UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COINT - TECNOLOGIA EM SISTEMAS PARA INTERNET
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

DANILO AUGUSTO PINOTTI DE MELLO

SOLUÇÃO PARA MONITORAMENTO AMBIENTE UTILIZANDO ARDUINO

TRABALHO DE CONCLUSÃO DE CURSO

GUARAPUAVA
2017

DANILO AUGUSTO PINOTTI DE MELLO

SOLUÇÃO PARA MONITORAMENTO AMBIENTE UTILIZANDO ARDUINO

Trabalho de Conclusão de Curso apresentado ao Curso de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Tecnólogo.

Orientador: Prof. Ms. Paulo Henrique Soares
UTFPR

Coorientador: Prof. Fábio Leandro Janiszevski
Unicentro

GUARAPUAVA
2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Guarapuava
Curso Superior de Tecnologia em Sistemas Para a Internet

ATA DE DEFESA DE MONOGRAFIA DE TRABALHO DE CONCLUSÃO DE CURSO DO CURSO DE TSI

No dia 14 de junho de 2017, às 16:00 horas, nas dependências da Universidade Tecnológica Federal do Paraná Câmpus Guarapuava, ocorreu a banca de **defesa da monografia** de Trabalho de Conclusão de Curso intitulada: "**Solução para Monitoramento Ambiente Utilizando Arduino**" do acadêmico **Danilo Augusto Pinotti De Mello** sob orientação do professor **Prof. Me. Paulo Henrique Soares** do Curso de Tecnologia em Sistemas para Internet.

Banca Avaliadora	
Membro	**Nome**
Orientador	Prof. Me. Paulo Henrique Soares
Coorientador	Prof. Esp. Fábio Leandro Janiszevski
Avaliador 1	Prof. Dr. Luciano Ogibolski
Avaliador 2	Prof. Esp. Maurício Barfknecht

Situação do Trabalho	
Situação	(x) Aprovado ( ) Aprovado com ressalvas ( ) Reprovado ( ) Não Compareceu
Encaminhamento do trabalho para biblioteca	(x) Pode ser encaminhado para biblioteca. ( ) Manter sigilo para publicação ou geração de patente.

Guarapuava, 14 de junho de 2017.

A Folha de Aprovação assinada encontra-se na Coordenação do Curso (ou Programa).

Dedico este trabalho a Deus e a todos os parentes, amigos e colegas pelo incentivo constante.

AGRADECIMENTOS

Primeiramente agradeço a Deus que iluminou meu caminho durante esta caminhada.

Devo um agradecimento especial ao meu professor coorientador, Fabio Leandro Janiszevski, que teve paciência e que me ajudou a concluir este trabalho, agradeço também aos meus professores que me ajudaram a construir toda a base para pode desenvolver este trabalho.

Agradeço o professor Hermano Pereira por me motivar e me ensinar, principalmente, a estabelecer prioridades sobre os acontecimentos na vida e a importância de Deus nela.

Agradeço a Universidade Tecnológica Federal do Paraná por prover todo o recurso necessário para auxiliar em meu desenvolvimento profissional e pessoal.

RESUMO

MELLO, Danilo A. P.. SOLUÇÃO PARA MONITORAMENTO AMBIENTE UTILIZANDO ARDUINO. 2017. 39 f. Trabalho de Conclusão de Curso – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2017.

Existem diversos ambientes em que é estritamente necessário o monitoramento, como por exemplo, centros de processamento de dados, câmaras frias e granjas.
Atualmente é possível encontrar sistemas desenvolvidos unicamente para esta finalidade, porém nem sempre estes estão ao alcance de todos devido a fatores como, custo de aquisição e implantação, ou disponibilidade no mercado nacional.
O objetivo deste projeto foi desenvolver um sistema utilizando a plataforma Arduino juntamente com sensores, de forma que o sistema alerte devidamente um usuário, ou um grupo de usuários em casos de anomalias no ambiente e que seja capaz de fornecer informações sobre cada ambiente monitorado através de uma interface Web.
Palavras-chave: Arduino, Sensores, MQTT, Pub-Sub, Monitoramento Ambiente.

ABSTRACT

MELLO, Danilo A. P.. Environment monitor solution with Arduino. 2017. 39 f. Trabalho de Conclusão de Curso – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2017.

There are several environments in which it is strictly necessary to monitor, such as data processing centers, cold chambers and grange.
Nowadays, it is possible to find systems developed solely for this purpose, but they are not always available to everyone due to factors such as acquisition and deployment cost or availability in the domestic market.
The objective of this project was to develop a system using the Arduino platform with sensors, so that the system duly warns a user or a group of users in cases of anomalies in the environment and that is able to provide information about each environment monitored through a Web interface.

Keywords: Arduino, Sensors, MQTT, Pub-Sub, Environment Monitor.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACL	Access Control List
API	Application Programming Interface
CPD	Centro de Processamento de Dados
CSS	Cascading Style Sheets
DB2	Database 2
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Server
HTML	Hypertext Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IBM	International Business Machines
IDE	Integrated Development Environment
IP	Internet Protocol
LCD	Liquid Crystal Display
LDR	Light Dependent Resistor
LED	Light Emitter Diode
M2M	Machine-to-Machine
MAC	Media Access Control
MIT	Massachusetts Institute of Technology
MQTT	Message Queueing Telemetry Transport
PHP	Hypertext Preprocessor
PRTG	Paessler Router Traffic Grapher
PWM	Pulse Width Modulation
QoS	Quality of Service

SGBD	Sistema Gerenciador de Banco de Dados
SMS	Short Message Service
SNMP	Simple Network Management Protocol
SPI	Serial Peripheral Interface
SQL	Structed Query Language
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TI	Tecnologia da Informação
UTFPR	Universidade Tecnológica Federal do Paraná
XML	EXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Milhares de dispositivos novos são conectados à Internet todos os dias. Segundo Evans (2011), entre 2003 e 2010, a quantidade de dispositivos conectados à Internet ultrapassaram a quantidade de pessoas no mundo. Pesquisas apontam que até o ano de 2020, a quantidade de dispositivos serão equivalentes a 6,5 vezes a quantidade de pessoas, não relacionada somente à dispositivos pessoais. Atualmente, cerca de 75% dos dispositivos conectados à Internet são para usos pessoais e, até 2020, este número tende a cair para 25%. Toda essa quantidade de dispositivos conectados pode ser relacionada com a Internet das Coisas. O termo Internet das Coisas é utilizado para definir o uso de sistemas embarcados conectados à Internet que possuem a capacidade de se comunicarem com outros dispositivos, serviços ou pessoas em escala global (MUKHOPADHYAY, 2014).

De acordo com Cunha (2007) , sistema embarcado é a capacidade computacional aplicada em um circuito integrado, equipamento ou sistema. Estes sistemas, por sua vez, são criados para exercer apenas uma funcionalidade durante sua vida útil, como fornos de micro-ondas, roteadores, televisores, dentre outros. Também são exemplos de sistemas embarcados sensores com sistemas operacionais próprios, sistemas expansíveis que trabalham com protocolos específicos de monitoramento ou até mesmo ares condicionados de precisão. Além disso, sistemas embarcados são utilizados como dispositivos para monitoramento ambiente, os quais verificam constantemente as condições atuais de temperatura e umidade para tomarem as devidas providências. Um exemplo é o uso de alarmes de incêndio que, quando disparados, notificam o corpo de bombeiros automaticamente para requisitar o envio de uma equipe profissional no local para averiguar a situação.

No decorrer do presente trabalho, foi desenvolvido um sistema que integra sensores através de um Arduino. Esta solução é capaz de realizar o monitoramento de dados de determinados ambientes, de forma que a mesma seja capaz de notificar grupos de usuários do sistema de acordo com gatilhos pré-programados, como por exemplo, uma variação brusca de temperatura, umidade e luminosidade.

1.1 OBJETIVOS

Serão apresentados nesta seção o objetivo geral e os objetivos específicos do projeto.

1.1.1 OBJETIVO GERAL

Desenvolver uma solução para monitoramento ambiente utilizando Arduino e sensores, de forma que notifique usuários do sistema em casos de anomalias na temperatura, luminosidade e umidade.

1.1.2 OBJETIVOS ESPECÍFICOS

- Implementar comunicação cliente-servidor com o uso do protocolo MQTT (*Message Queueing Telemetry Transport*).
- Implementar em Arduinos, sensores que verifiquem as condições do ambiente como luz, temperatura e umidade.
- Desenvolver um sistema Web que integre os Arduinos com sensores.
- Integrar ferramentas de notificações no sistema Web.
- Monitorar o ambiente real e confirmar com dados dos sensores que serão implantados no CPD (Centro de Processamento de Dados) da UTFPR (Universidade Tecnológica Federal do Paraná) câmpus Guarapuava para validação do sistema desenvolvido.

1.2 DIFERENCIAL TECNOLÓGICO

Destaca-se que no projeto é possível monitorar mais de um ambiente ao mesmo tempo apenas incluindo outro módulo (constituído por um Arduino, sensores e *shield* Ethernet) no sistema e instalando-o fisicamente. Além disso, foram utilizados para o projeto apenas componentes de baixo custo e tecnologias de desenvolvimento *open-source*, possibilitando desta forma a implantação sem um alto investimento.

2 REVISÃO DE LITERATURA

Nesta seção são abordadas as tecnologias utilizadas no projeto.

2.1 HTML

O HTML, abreviação de *Hypertext Markup Language*, é uma linguagem de marcação utilizada no desenvolvimento de sites. Criada em 1991 por Tim Berberners-Lee, esta surgiu junto com o protocolo HTTP. Foi a primeira linguagem de nível mundial, porém, não foi a única. É possível criar páginas para a Web utilizando HTML mesclado com diversas outras linguagens, como o PHP (*Hypertext Preprocessor*), Ruby, dentre outros (PACIEVITCH, 2016).

A criação de códigos HTML é baseada em marcações conhecidas como *tags*, em que cada uma tem uma função diferente. Arquivos HTML podem ser criados e editados usando simples editores de texto e devem ser salvos com a extensão .htm ou .html.

A cada nova versão lançada a linguagem se torna mais fácil de ser interpretada e utilizada. Para o projeto foi utilizada a versão 5 que possui, nativamente, elementos para criação de gráficos (canvas e svg), elementos multimídia (áudio e vídeo) e novos elementos pré-definidos para formulários (W3SCHOOLS, 2016b).

2.2 CSS

O CSS, abreviação de *Cascading Style Sheets*, é uma linguagem utilizada para descrever estilos em documentos, sendo eles HTML ou XML (*EXtensible Markup Language*) (W3SCHOOLS, 2016a). É de responsabilidade do CSS definir o que será renderizado na tela ou não e de qual forma cada elemento será apresentado.

O CSS trabalha com o efeito cascata em que é estabelecida uma prioridade na aplicação das regras de estilo aos elementos. A prioridade é dada da seguinte forma (considerando o primeiro item como prioridade mais alta): estilos declarados no elemento, estilos declarados em uma folha de estilização local (no próprio arquivo HTML) e estilos declarados em folhas de estilização externas (arquivos com extensão .css). Outro fator que determina a prioridade é a ordem em que são declarados nos arquivos, quanto mais "próximo" do elemento e mais para baixo na declaração, maior a prioridade (SILVA, 2003).

2.3 PHP

Criado em 1994 por Rasmus Lerdorf, o PHP, inicialmente significando Personal Home Page, tinha como proposta utilizar os programas desenvolvidos pelo próprio criador em suas páginas Web. Com o passar do tempo e com a aceitação das pessoas, a linguagem foi tomando

forma até se tornar o que conhecemos hoje e com o significado de *Hypertext Preprocessor*, ou, pré-processador de hipertexto (ALVAREZ, 2004).

O PHP é uma linguagem de programação interpretada, amplamente utilizada para desenvolvimento Web, que devido a sua capacidade de se misturar ao HTML, torna mais fácil a criação de páginas dinâmicas (SOARES, 2010).

Reyes (2009) cita várias finalidades para o uso do PHP, dentre elas estão:

- *E-Commerce* através do Magento, Zen Cart, Shopify, dentre outros;
- Fóruns através do phpBB, vBulletin, PunBB, dentre outros;
- Aplicações para o Facebook;
- Sistemas para manuseio de conteúdo, como Wordpress, Joomla e Drupal.

De acordo com a Fuctura (2016), são vantagens em utilizar o PHP:

- A facilidade para aprender a linguagem;
- A praticidade para acessar SGBDs (Sistema Gerenciador de Banco de Dados), como o MySQL e o Oracle;
- O fato de ser multiplataforma permite que aplicações feitas em PHP funcionem em computadores com diversos sistemas operacionais diferentes;
- O PHP é *open-source*, ou seja, têm o código fonte aberto.

Também deve ser levado em consideração as desvantagens em utilizar a linguagem (FUCTURA, 2016):

- A compatibilidade entre versões não é totalmente confiável, ou seja, não há uma padronização de determinados recursos entre as versões;
- A segurança deixa a desejar quando comparada a um Servlet do Java EE, por exemplo.

De acordo com um *ranking* de linguagens mais utilizadas detalhado por Tiobe (2016) através de motores de busca, o PHP está em 6º lugar como linguagem de programação mais utilizada no mundo em 2016, como apresentado na Figura 1.

Este projeto usou o PHP na versão 7.0 como linguagem padrão para o sistema Web desenvolvido. Os motivos para esta escolha são a sua ampla comunidade de desenvolvedores e o suporte multiplataforma.

2.4 LARAVEL - FRAMEWORK PHP

Criado em 2011 por Taylor Otwell, o Laravel é um *framework* PHP *open-source* sobre a licença MIT (*Massachusetts Institute of Technology*), voltado para aplicações Web. Em cinco anos de existência, o Laravel se tornou um dos mais conhecidos *Frameworks* PHP e apontado como um dos melhores e mais completos (SCHIMIGUEL, 2016).

O Laravel utiliza uma *Engine* de *template* chamada de *Blade* para a criação da interface gráfica. Esta *Engine* traz uma série de ferramentas para auxiliar na criação rápida de interfaces agradáveis e funcionais, além da redução de códigos duplicados. (ADRIEL, 2015).

A comunicação com o Banco de Dados é feita através do Eloquent ORM. Esta comunicação é facilitada para o desenvolvedor visto que a ferramenta traz diversos recursos

May 2016	May 2015	Change	Programming Language	Ratings	Change
1	1		Java	20.956%	+4.09%
2	2		C	13.223%	-3.62%
3	3		C++	6.698%	-1.18%
4	5	^	C#	4.481%	-0.78%
5	6	^	Python	3.789%	+0.06%
6	**9**	^	**PHP**	**2.992%**	**+0.27%**
7	7		JavaScript	2.340%	-0.79%
8	15	^^	Ruby	2.338%	+1.07%
9	11	^	Perl	2.326%	+0.51%
10	8	v	Visual Basic .NET	2.325%	-0.64%

Figura 1 – Ranking das linguagens mais utilizadas

Fonte: Tiobe (2016).

para o gerenciamento das informações no Banco de Dados (ADRIEL, 2015).

Quando comparado com outros *frameworks*, o Laravel tem como desvantagem o desempenho, o tamanho relativamente maior, o tempo elevado de *debug* em determinadas ocasiões e a comunidade relativamente menor (AGRIYA, 2015), porém, a facilidade de aprendizado, os recursos, a documentação e a sua estabilidade fazem com que o Laravel seja um dos *frameworks* PHP mais utilizados no mundo e também no presente projeto.

2.5 SQL

A Linguagem Estruturada de Consultas (SQL, *Structed Query Language*) foi desenvolvida para ser capaz de criar e dar manutenção em bases de dados relacionais além de permitir o gerenciamento dos dados nessas bases (SHELDON, 2009).

Atualmente, boa parte dos SGBDs (sistemas gerenciadores de banco de dados) utilizam o SQL como linguagem padrão para o acesso às bases de dados. Dentre eles podem ser citados o DB2 da IBM, Oracle da Oracle Corporation, SQL Server da Microsoft, e MySQL da Oracle Corporation (XDSOFTWARE, 2016).

Mediante o SQL, o utilizador torna-se capaz de criar, apagar e modificar base de dados além de poder inserir, apagar, atualizar e buscar informações dos mesmos.

Dentre as utilidades do SQL citadas em TutorialsPoint (2016) do SQL podem ser destacadas que:

- Permite o usuário acessar dados em bases de dados relacionais;
- Permite a manipulação dos dados definidos na base de dados;
- Permite a criação e alteração de tabelas e bases de dados;
- Permite a embarcação com outras linguagens através da utilização de módulos, pré-compiladores e bibliotecas;

- Permite a criação de *views*, *stored procedures* e *functions* nas bases de dados.
- Permite aos usuários definirem permissões de acesso às tabelas, *procedures* e *views*.

O projeto utiliza o SQL como linguagem padrão para comunicação com o SGBD definido, o MySQL.

2.6 MySQL

O MySQL é um sistema gerenciador de banco de dados relacional, *open-source* e é um dos mais utilizados e conhecidos no cenário mundial. Criado na década de 90 por David Axmark, Allan Larsson e Michael Monty Widenius. O MySQL acabou se tornando conhecido devido ao seu desempenho ao gerenciar dados e, cada vez mais vem sendo utilizado em projetos e sistemas (MILANI, 2007).

Grandes empresas como Youtube, PayPal, Google, Facebook, Cisco, entre outras, utilizam o MySQL em seus sistemas (MYSQL, 2016).

De acordo com Gilmore (2008) a popularidade do MySQL foi conquistada devido aos seguintes motivos:

- Flexibilidade: Independentemente do sistema operacional que está sendo utilizado, o MySQL têm grandes chances de atender à necessidade.
- Capacidade: Desde o início do projeto, o foco do MySQL é o desempenho. Com as atualizações e facilidades empregadas, a equipe de desenvolvimento ainda mantém um compromisso em relação à sua velocidade.
- Recursos SQL a nível de empresa: A partir da versão 5.0 o MySQL traz suporte a views, subqueries e stored procedures.
- Indexação e busca *full-text*: São recursos que permitem maior velocidade em buscas de forma a trazer os resultados por ordem de relevância.
- *Query caching*: Permite que o MySQL armazene em memória os resultados das buscas junto com a *querie SELECT* (recurso utilizado para buscar informações na base de dados).
- Replicação: Permite que um banco de dados seja replicado em um ou mais servidores MySQL.
- Segurança: O MySQL permite, através de configurações, definir permissões de acesso para diversos recursos e configurações para praticamente tudo que o SGBD é capaz de disponibilizar.
- Opções flexíveis para licença além da sua comunidade de usuários altamente ativa.

Por ser um SGBD livre, eficaz e com uma comunidade relativamente ativa, o MySQL foi escolhido para ser utilizado no projeto para armazenar dados no sistema.

2.7 ARDUINO

O Arduino é uma plataforma eletrônica *open-source* de prototipagem baseada em hardware e software flexíveis e fáceis de usar (ARDUINO, 2016g). São chamados de projetos *open-source* aqueles em que qualquer pessoa pode modificar e distribuir, tornando-o desta forma, totalmente acessível ao público (OPENSOURCE, 2016).

Segundo McRoberts (2010), o Arduino é um pequeno computador onde pode ser programado para processar entradas e saídas entre o dispositivo e os componentes externos conectados a ele, como por exemplo, ligar um ventilador que está conectado a um relé quando a temperatura do ambiente for superior a 30°C.

Existem diversos modelos de Arduino disponíveis para a utilização, dentre eles estão:

- Arduino Uno: dado como um dos mais simples devido à sua baixa capacidade de armazenamento e processamento. É ideal para quem está iniciando na área de eletrônica e programação (ARDUINO, 2016e).
- Arduino Mega: possui uma quantidade maior de portas analógicas e digitais e maior capacidade de processamento e armazenamento em relação ao modelo Uno. Este, por sua vez é utilizado em projetos maiores que demandam maior capacidade do dispositivo (ARDUINO, 2016b).
- Arduino Yún: placa projetada para Internet das Coisas, onde visa principalmente a conectividade entre dispositivos. Combina a capacidade de um Linux com a facilidade de um Arduino (ARDUINO, 2016f).

A programação do Arduino é feita através de um ambiente de desenvolvimento próprio, onde utiliza-se de uma linguagem de programação também própria, semelhante ao C/C++.

Arduinos normalmente são utilizados em projetos para automação industrial, ou residencial, e podem ser conectados a displays, LEDs (*Light Emitter Diode*), sensores, motores, botões e qualquer outro dispositivo que possa ser controlado ou que emita dados.

2.7.1 ARDUINO MEGA

O Arduino Mega, apresentado na Figura 2, foi projetado para executar projetos mais complexos e que exijam mais capacidade de processamento, armazenamento e quantidade de pinos analógicos e digitais. Este possui 54 pinos digitais de entrada e saída e mais 16 pinos analógicos (ARDUINO, 2016b).

A Tabela 1 apresenta informações adicionais sobre o Arduino Mega.

Os motivos pelos quais foi escolhido o Arduino Mega para o projeto são: baixo custo, capacidade para alimentação externa, portas analógicas e digitais suficientes para possíveis expansões futuras.

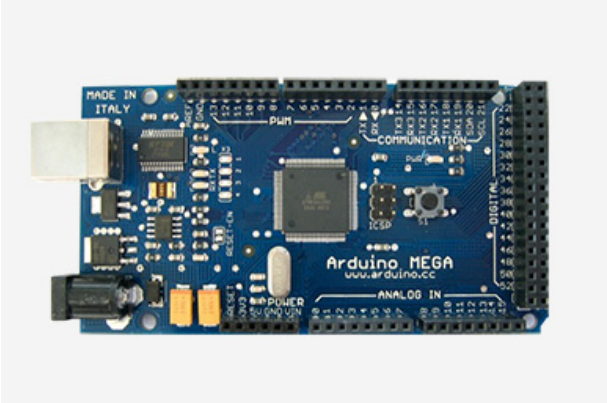

Figura 2 – Arduino Mega

Fonte: Arduino (2016).

Tabela 1 – Características do Arduino Mega

Tensão de operação	5V
Tensão de entrada (recomendado)	7-12V
Tensão de entrada (limites)	6-20V
Memória flash	128KB com 4KB utilizado no bootloader
SRAM	8KB
EEPROM	4KB
Velocidade de Clock	16MHz
Quantidade de portas digitais	54 (onde 15 fornecem saída PWM)
Quantidade de portas analógicas	16
Corrente contínua por pino	40mA
Corrente contínua do pino 3.3V	50mA

Fonte: Adaptada de Arduino (2016).

2.7.2 MÓDULOS

Módulos são dispositivos que permitem aumentar a capacidade do Arduino. Estes são conectados nas portas digitais e/ou analógicas separadamente e através de fios (LIMA, 2013). Os módulos ocupam as portas disponíveis no Arduino, ou seja, terão que ser destinadas portas analógicas e/ou digitais exclusivamente para o módulo de acordo com a necessidade do mesmo.

Existem módulos para as mais diversas funcionalidades, por exemplo, módulo relé, módulo Bluetooth, módulo Ethernet (Figura 3), dentre outros.

2.7.3 SHIELDS

Shields são placas que podem ser conectadas por cima do Arduino e que estendem sua capacidade (ARDUINO, 2016c). Existe uma variedade grande de *shields* para Arduino, como *shield* com tela LCD (*Liquid Crystal Display*), *shield* para leitura de cartões microSD, dentre outros.

A principal vantagem em utilizar *shields* é que, ao contrário dos módulos, mesmo com elas instaladas, as portas digitais e analógicas nem sempre são ocupadas. Existem modelos

Figura 3 – Módulo Ethernet para Arduino

Fonte: Filipeflop (2016).

de shields que conseguem manter todas as portas (*shield* Ethernet por exemplo) e também modelos que ocupam todas as portas (alguns modelos de *shield* LCD por exemplo).

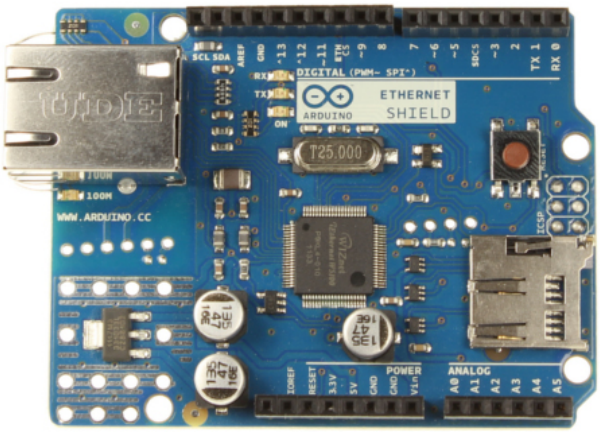

Figura 4 – Shield Ethernet para Arduino

Fonte: Arduino (2016).

Para o presente projeto será utilizado uma *shield* Ethernet (Figura 4) que irá conectar um Arduino em uma rede local ou à Internet utilizando um conector RJ45. O *shield* em questão trabalhará sob o protocolo TCP/IP com velocidade de conexão de 10/100Mb e será alimentado com 5V pela própria placa Arduino. Sua comunicação com o Arduino é dada através da porta SPI (*Serial Peripheral Interface*), desta forma, não consumindo as outras portas da placa (ARDUINO, 2016a).

2.8 SENSORES

Sensores são dispositivos eletroeletrônicos que têm a propriedade de transformar uma grandeza física em sinais elétricos através de uma ou mais propriedades do material de que é feito o sensor (STEFFENS, 2016).

Existem sensores para as mais diversas necessidades como medir temperatura, medir a concentração de gás carbônico, medir a luminosidade, detectar som, detectar movimento, dentre outros. A Figura 5 apresenta alguns exemplos de sensores como sensor de infr-avermelho, sensor de umidade, entre outros.

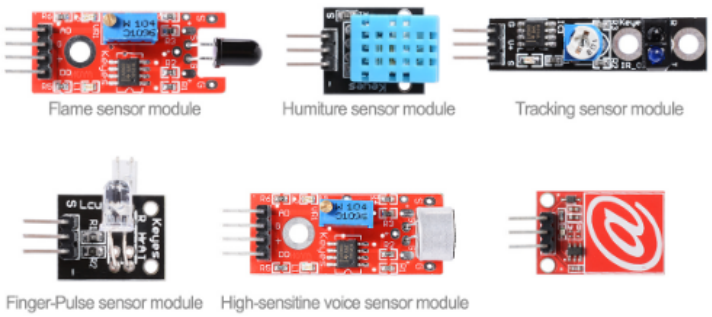

Figura 5 – Exemplos de sensores

Fonte: Adaptada de AliExpress (2016).

Todos os sensores podem ser classificados, basicamente, em dois tipos, podendo eles serem sensores analógicos ou sensores digitais. Esta divisão é feita de acordo com a resposta do sensor à variação da condição. Sensores analógicos são os que retornam, mesmo que em uma faixa de valores definida, resultados infinitos intermediários a esta faixa, já os sensores digitais, retornam apenas dois resultados, sendo ele verdadeiro ou falso (LOW ou HIGH no Arduino) (PARSKO, 2006).

2.8.1 LDR

O LDR (*Light Dependent Resistor*) é um componente eletrônico capaz de variar sua resistência elétrica em função da luminosidade incidente sobre ele. A resistência deste sensor é inversamente proporcional à quantidade de luz sobre ele, ou seja, quanto mais luz, menos resistência. O cálculo da resistência de um LDR é dado obedecendo a seguinte equação:

$$R = C.L.a$$

De forma que L representa a luminosidade em Lux, "C" e "a" constantes individuais de acordo com o material utilizado e do processo de fabricação (BRITTO, 2003).

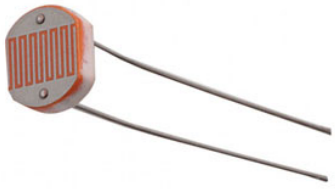

Figura 6 – Sensor LDR

Fonte: NinjaGecko (2016).

Um LDR comum (Figura 6) tem aproximadamente a resistência de 5.000 ohms sob a luz do dia e 20.000.000 ohms na escuridão (KITRONIK, 2014). A Figura 7 apresenta o gráfico da relação entre a intensidade da luz e a resistência de um LDR.

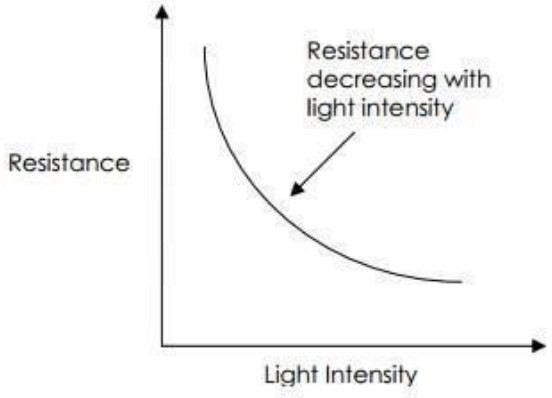

Figura 7 – Gráfico da resistência de um LDR em função da luminosidade

Fonte: Eletrical4U (2016).

Por ter um preço baixo e estrutura simples, o LDR é utilizado, por exemplo, em postes de luz, despertadores, alarmes, contagem de pacotes em esteiras, dentre outros (ELETRICAL4U, 2016).

Além disso, o sensor também é utilizado em circuitos para detectar luminosidade em ambientes abertos ou fechados, podendo desta forma, detectar em ambientes sem iluminação um possível incêndio, a abertura de uma porta ou a passagem de pessoas em determinadas ocasiões. Devido a estas características, este sensor é um dos componentes presente neste projeto.

2.8.2 LM35

Criado pela National Semiconductor, a série LM35 é formada por dispositivos com circuitos integrados de precisão com o objetivo de obter a temperatura de ambientes. A temperatura é obtida através da tensão de saída do sensor e, esta é diretamente proporcional à temperatura do ambiente onde o sensor está localizado (NATIONALSEMICONDUCTOR, 2000).

A Figura 8 apresenta um LM35 (lado esquerdo) junto com sua representação gráfica (lado direito).

De acordo com o *Datasheet* do sensor, o LM35 é capaz de trabalhar com tensão de entrada entre 4 e 30 volts e trabalha corretamente em uma faixa de temperatura de -55°C até 150°C.

A precisão aprimorada quando comparada ao DHT11 e o custo relativamente baixo deste sensor são motivos dele ser utilizado no projeto.

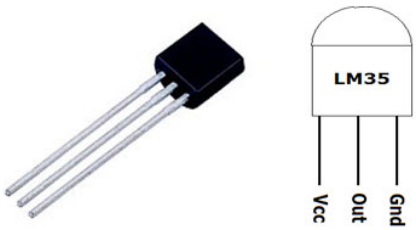

Figura 8 – Sensor LM35

Fonte: Saddam (2015).

2.8.3 DHT11

O DHT11 (Figura 9) é um sensor digital capaz de obter a temperatura e umidade do ambiente em que está presente. O DHT11 não é um sensor de alta precisão, logo, não é recomendado para ambientes de alto risco (ARDUINOECIA, 2013).

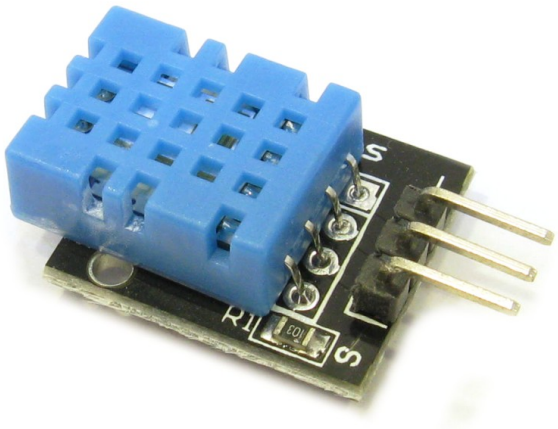

Figura 9 – Sensor DHT11

Fonte: Modtronix (2016).

Este sensor trabalha com a tensão elétrica entre 3.3V e 5.5V, com a faixa de umidade entre 20 e 90% e faixa de temperatura entre 0 e 50°C. A precisão do sensor é de 2°C e 5% de umidade. Das características do DHT11, pode ser destacado o baixo consumo de energia, sinal digital calibrado, o fato de não necessitar de mais componentes para poder usar e às bibliotecas disponíveis para suporte (SUNROM, 2012).

O DHT11 é importante para o projeto para a captação da umidade do ambiente visto que é um dos sensores de umidade mais baratos encontrados no mercado e devido a facilidade de uso na aplicação.

2.9 PROTOCOLO MQTT

Criado por Andy Stanford-Clark, com suas origens na IBM, o MQTT, *Message Queueing Telemetry Transport*, é um protocolo de comunicação M2M (*Machine-to-Machine*)

para dispositivos dentro de um domínio *publish/subscribe* projetado para Internet das Coisas, desta forma, é caracterizado pela sua comunicação rápida e leve (BROWN, 2014).

Os padrões do protocolo se encontram atualmente na versão 3.1. De acordo com suas especificações, ele foi projetado para ser *open-source*, simples, leve e fácil de implementar. Essas características tornam o protocolo ideal para uso em redes congestionadas, grandes, de baixa velocidade ou não seguras. Além disso, o protocolo é ideal para ser utilizado em dispositivos embarcados cuja memória e capacidade de processamento são limitadas (EUROTECH, 2010).

O protocolo utiliza do padrão de envio de mensagens *publish/subscribe* o qual torna possível o envio de mensagens de um para vários dispositivos (um-para-muitos). Também são características do protocolo (LEE et al., 2013):

- Conexão com a rede através do TCP/IP;
- Com o cabeçalho de apenas 2 bytes, o MQTT provê trocas minimizadas de informações resultando em menos tráfego na rede;
- Assíncrono com três níveis de qualidade de serviço (QoS):
  - "No máximo uma vez": nível utilizado onde as mensagens são enviadas apenas uma vez pela rede, desta forma, correndo o risco de se perderem no meio do caminho. Um exemplo de uso seria o de envio de dados de sensores onde, caso perca algum dos valores, logo em seguida será enviado outro;
  - "Ao menos uma vez": nível onde as mesmas mensagens são enviadas mais de uma vez pela rede. Usando esta configuração, pode acontecer da mensagem chegar duplicada no destinatário;
  - "Exatamente uma vez": nível em que garante a chegada da mensagem ao destinatário. Utilizada quando a informação é crítica e precisa chegar sem erros ou duplicações ao destinatário, exemplo, uma transação monetária.

O *Facebook Messenger* é um exemplo de produto criado com base no MQTT (LEE et al., 2013). Este protocolo é o responsável no projeto pela comunicação entre o cliente e o servidor devido a simplicidade e desempenho em redes congestionadas ou lentas.

2.10 PADRÃO PUBLISH/SUBSCRIBE

O princípio do modelo de comunicação *Publish/Subscribe* (pub/sub) está em os dispositivos registrarem seus interesses nos tipos de informações para que possam recebê-las, sempre que possível, dos dispositivos fornecedores. O processo de registrar tal interesse é chamado de subscrição, portanto, o dispositivo interessado logo é chamado de subscritor (*subscribers*). Por outro lado, os dispositivos que irão fornecer as mensagens passarão por um processo chamado de publicação, Estes são chamados de publicadores (*publishers*) (HUNKELER; TRUONG; STANFORD-CLARK, 2008).

De acordo com Hunkeler e Truong (2008) existem três tipos de sistemas pub/sub:

- Baseado em tópicos: neste tipo de sistema, todos os tópicos já estão definidos do inicío ao final do processo de desenvolvimento.

- Baseado em tipos: os subscritores declaram qual tipo de dado tem interesse (dados de temperatura por exemplo).
- Baseado no conteúdo: este é o tipo mais versátil dentre os outros. O subscritor declara qual o conteúdo que desejará receber (pode haver dados de sensores de luz e temperatura ao mesmo tempo).

Um sistema *publish/subscribe* é implementado sobre um agente de eventos conhecido como *broker*. *Brokers* são responsáveis em rotear eventos entre os publicadores e os subscritores (BANAVAR et al., 1999). A Figura 10 faz uma representação da arquitetura do padrão em questão.

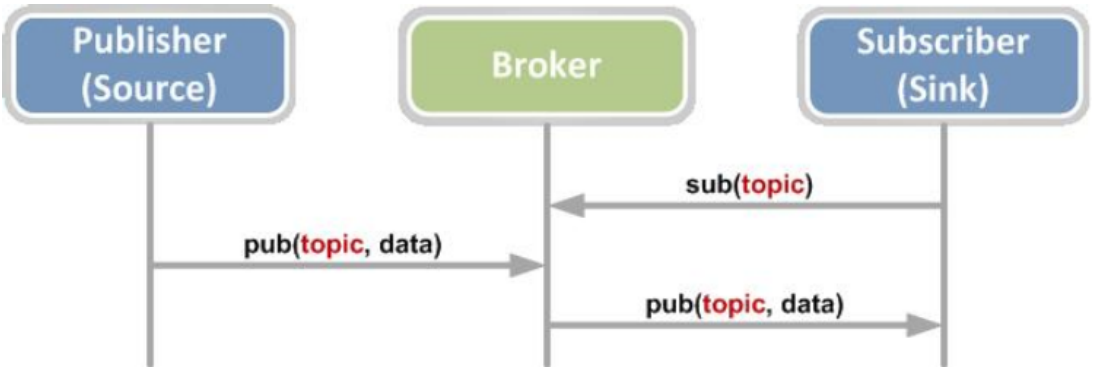

Figura 10 – Representação da arquitetura Pub/Sub

Fonte: Adaptada de Kannan (2015).

O protocolo MQTT, utilizado no projeto, é implementado no padrão *publish/subscribe*, logo, a comunicação entre o módulo Arduino e o servidor irão respeitar as especificações recém descritas.

2.11 TELEGRAM

Telegram é um aplicativo *open-source* de mensagem para desktop e mobile baseado em nuvem, fundado em 2013, cujo foco é sua segurança e velocidade. É possível utilizar o Telegram em vários dispositivos ao mesmo tempo, e sua sincronização de mensagens é dita como ideal, independentemente da quantidade de clientes conectados a uma mesma conta (TELEGRAM, 2016b).

A encriptação das mensagens no Telegram, seja texto, foto, áudio, vídeo, dentre outros, pode ser feita em duas camadas. A primeira delas é entre o servidor e o cliente, utilizada em chats privados ou em grupos de conversa, e, a segunda, além da camada de cliente-servidor, é feita também uma outra entre remetente e destinatário (cliente para cliente) (TELEGRAM, 2016b).

Além da segurança e velocidade do Telegram, é possível a utilização e criação de Bots através da API (*Application Programming Interface*) aberta de Bots.

De acordo com a equipe de desenvolvimento do Telegram (2016), Bots são aplicações de terceiros que funcionam dentro do Telegram. Usuários podem interagir com estes Bots através de mensagens e linhas de comando. Bots são controlados através de requisições HTTPS (*Hyper Text Transfer Protocol Secure*) para a API de Bot. A Figura 11 apresenta um exemplo de Bot cuja função programada para ele é de enviar materiais de estudo para os usuários poderem aprender línguas novas.

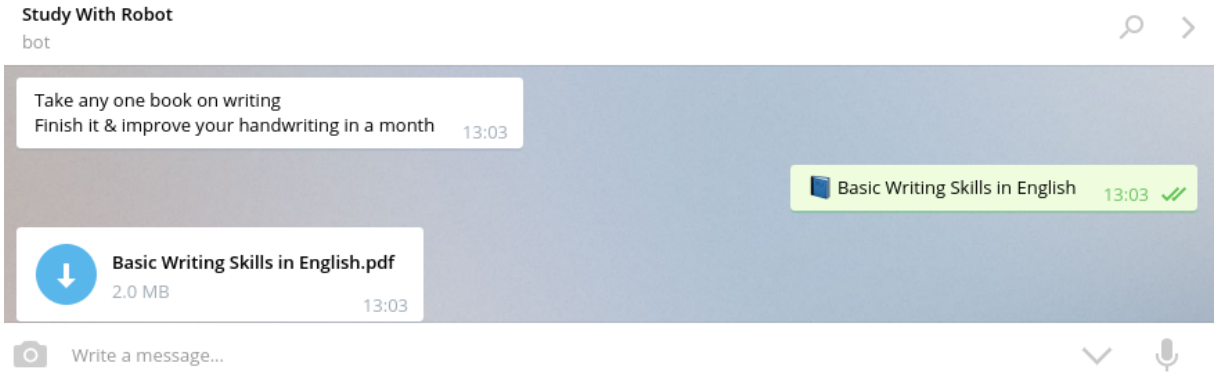

Figura 11 – Exemplo de Bot do Telegram.

Fonte: O autor

O Telegram é um dos meios utilizados para alertar os usuários de possíveis incidentes nos ambientes monitorados através de um Bot criado para tal finalidade.

2.12 UTILIZAÇÃO DAS TECNOLOGIAS NA SOLUÇÃO

Cada tecnologia apresentada neste capítulo têm participação importante no sistema.

A Tabela 2 apresenta, resumidamente, a utilização das tecnologias na solução desenvolvida.

Tabela 2 – Uso das tecnologias apresentadas

Tecnologia	Descrição
HTML e CSS	Utilizados na interface gráfica do sistema juntamente com Javascript e suas bibliotecas.
Laravel Framework	Utilizado como base para o servidor central da solução desenvolvida.
MySQL	Utilizado como sistema gerenciador de banco de dados para armazenar os dados vindos dos sensores e informações do sistema.
Arduino	Utilizado para reunir os sensores e enviar as informações para o servidor central.
Arduino Ethernet Shield	Utilizado para prover ao Arduino uma conexão com a rede local.
Sensores	Utilizados para obter informações do meio em que estão instalados.
Protocolo MQTT	Utilizado como forma de comunicação entre o Arduino e o servidor central.
Telegram BOT	Um dos meios utilizados para notificar os usuários.

A solução apresentada no Capítulo 5, foi desenvolvida com a utilização de todas as tecnologias apresentadas neste capítulo anteriormente.

3 METODOLOGIA

Neste capítulo são apresentadas as etapas realizadas a fim de alcançar os objetivos do projeto.

- Definição das tecnologias;
- Estudo das tecnologias definidas;
- Desenvolvimento do sistema Web;
- Realização de testes em um CPD real.

3.1 DEFINIÇÃO DAS TECNOLOGIAS

Inicialmente, foram definidas as tecnologias a serem utilizadas no projeto. São elas: HTML, CSS, PHP, MySQL, Laravel, Apache Apollo, Arduino, LM35, LDR e DHT11. O desenvolvimento foi realizado dentro dos limites impostos por estas tecnologias. Portanto, foi importante escolher quais melhor atendem as necessidades especificadas. Para a escolha das tecnologias foram avaliados critérios como a eficácia, a comunidade relacionada e o custo de aquisição e manutenção de cada uma.

3.2 ESTUDO DAS TECNOLOGIAS DEFINIDAS

Esta etapa foi constituída por pesquisas, na Internet e em livros referentes às tecnologias utilizadas no projeto, até que se estabeleceu uma base de conhecimento necessária para dar início ao processo de desenvolvimento. As fontes de informação mais utilizadas durante o desenvolvimento do projeto foram: documentação oficial do Laravel[1], documentação oficial do Apache Apollo[2] e a documentação oficial do PHP[3].

3.3 DESENVOLVIMENTO DO SISTEMA WEB

Mediante o Arduino IDE (*Integrated Development Environment*) e um editor de texto voltado para programação, foi desenvolvido o sistema Web responsável em prover uma interface para o usuário. Também foi feito o algoritmo base do Arduino, este responsável por obter os dados dos sensores e enviar para o servidor Web.

3.4 REALIZAÇÃO DE TESTES EM UM CPD REAL

Após a conclusão do desenvolvimento, o sistema passou por uma etapa de implantação no CPD da UTFPR câmpus Guarapuava e, até a entrega do presente documento, passa por

[1]Documentação oficial do Laravel 5.3: https://laravel.com/docs/5.3/
[2]Documentação oficial do Apache Apollo: https://activemq.apache.org/apollo/
[3]Documentação oficial do PHP: https://secure.php.net/

observações. Nesta etapa, o sistema já trabalha com dados reais e, é de importante função, a comparação dos dados processados por ele com os dados reais obtidos através de outras ferramentas do gênero, como termômetros e câmeras.

4 DESENVOLVIMENTO

Nesta seção serão descritas as etapas e características do desenvolvimento do projeto.

4.1 CENÁRIO DE BASE

O sistema foi desenvolvido, tomando como base para o projeto, o CPD (Centro de Processamento de Dados) da UTFPR câmpus Guarapuava, onde possui dois aparelhos de ares condicionados e é definido, pela equipe responsável pelo CPD, um limite máximo de temperatura de 25°C. Tal limite foi imposto de acordo com as normas descritas em TIA-942 onde a temperatura ideal de um CPD deve ser entre 20 e 25°C.

O ambiente em questão forneceu todos os recursos necessários para a realização do projeto, tal como, energia elétrica, espaço físico para instalação do módulo e cada sensor dele, acesso à rede interna e acesso à Internet.

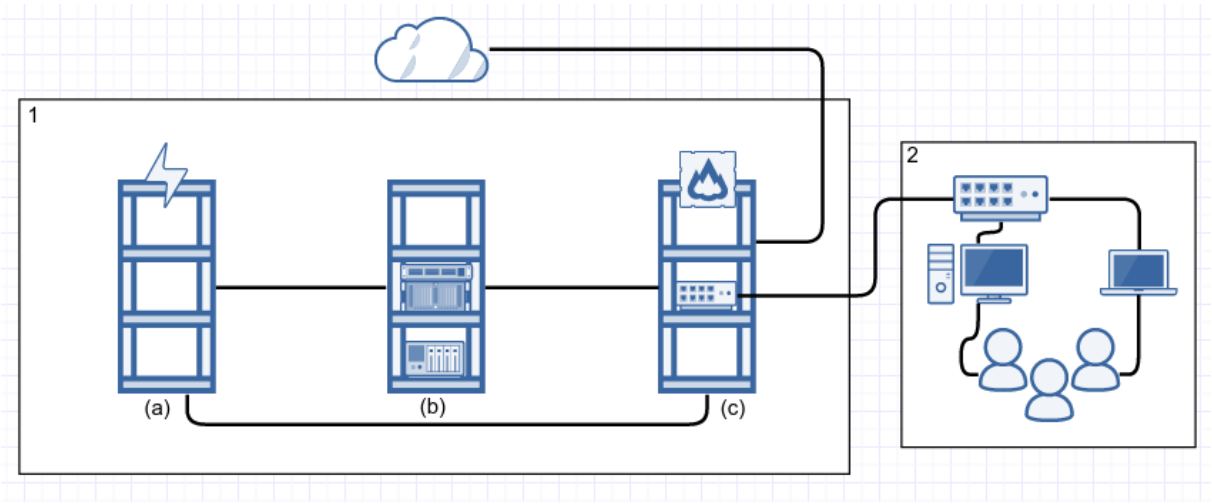

Figura 12 – Cenário proposto

Fonte: O autor

A Figura 12 retrata o cenário descrito. O ambiente representado pelo número 1 na Figura descreve o CPD em que (a) indica o rack de energia, (b) o rack de servidores, onde ficará o servidor central do sistema, e (c) o rack de rede. A instalação do sistema será feita no ambiente 1 e, através da rede, o módulo Arduino será diretamente conectado à rede por meio de um *switch* localizado no rack de rede (c). O ambiente 2 representa o local onde se encontra equipe de TI, responsável pelo CPD e pelos serviços de rede oferecidos pelo câmpus.

4.2 MATERIAIS UTILIZADOS

Nesta seção serão apresentados os materiais utilizados para o desenvolvimento do projeto.

4.2.1 HARDWARE

Para o desenvolvimento do projeto foi necessário:

- Um Arduino Mega: Utilizado para integrar todos os sensores e enviar as informações para o servidor;
- Uma *shield* Ethernet: Utilizada para dar ao Arduino Mega acesso à rede local através de um conector RJ45 e sob o protocolo TCP/IP;
- Sensores: Sensores DHT11, LM35 e LDR são responsáveis por captar dados sobre o ambiente em tempo real;
- Estrutura de rede: É necessário ter uma rede local funcional com rotas definidas entre o Arduino e o servidor Web para que possa haver comunicação entre eles;
- Servidor Web: O servidor Web é usado para armazenar os dados obtidos e prover uma interface para os usuários.

4.2.2 SOFTWARE

Nesta seção serão descritas as ferramentas utilizadas para o desenvolvimento e funcionamento do sistema.

4.2.2.1 ARDUINO SOFTWARE

O Arduino Software (apresentado na Figura 13) é uma IDE *open-source* que torna mais fácil a escrita de códigos e o envio do mesmo para o Arduino. Desenvolvido sob a linguagem Java, este é multiplataforma, ou seja, é possível utilizá-lo em diversos sistemas operacionais, como Mac OS X, Windows e Linux (ARDUINO, 2016d).

A utilização do Arduino Software se tornou essencial no projeto pois, através dele, é possível escrever o código, fazer verificação de sintaxe, compilar o código escrito, enviar o algoritmo compilado para o Arduino e fazer monitoramento Serial. Será utilizada a versão mais recente desta IDE, atualmente, encontrada na 1.6.9.

A linguagem de programação utilizada para programar em um Arduino foi feita baseada em C/C++ trazendo algumas modificações e alguns recursos nativos essenciais, como a função utilizada para ler valores em portas digitais e analógicas também como constantes nativas como HIGH e LOW ou INPUT e OUTPUT (ARDUINO, 2016h).

A IDE possui uma gama de exemplos de uso de componentes e *shields* que podem ser utilizados como base para desenvolvimento para outros projetos.

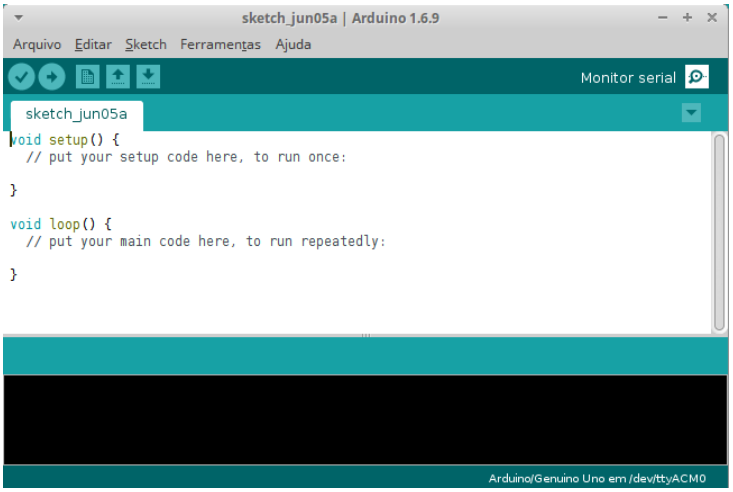

Figura 13 – Arduino Software

Fonte: O autor

4.2.2.2 ACTIVEMQ APOLLO

Diversos protocolos de comunicação necessitam de uma camada intermediária entre o dispositivo remetente e o destinatário para que seja possível o envio e recebimento das mensagens. Essas camadas intermediárias são chamadas de *brokers*.

O ActiveMQ Apollo é um projeto *open-source*, confiável e rápido que funciona como *broker* para diversos protocolos, como o MQTT, SSL, STOMP, etc. (ACTIVEMQ, 2016).

Foi utilizado no projeto o protocolo MQTT para a comunicação entre máquinas (M2M). O Apollo atua como *broker* da conexão entre clientes MQTT que é um protocolo *open-source* utilizado em dispositivos com recursos limitados e redes muito congestionadas usando um domínio *publish-subscribe* (ACTIVEMQ, 2016).

4.3 ARQUITETURA DO SISTEMA PROPOSTO

A solução consiste em duas partes. A primeira se trata de um servidor central que reunirá informações dos ambientes monitorados, e dispara notificações em ocasiões definidas por usuários.

A segunda parte consiste em um Arduino conectado a sensores de luminosidade, sensores de temperatura e sensores de umidade juntamente com uma *shield* Ethernet conectada à Internet ou a uma rede local, a qual é responsável por enviar todas as informações recolhidas do ambiente para o servidor. Para cada ambiente monitorado deve haver um ou mais módulos Arduino.

A Figura 14 retrata a arquitetura física do sistema que possui 4 salas (cada uma dentro de um retângulo isolado). A sala representada pelo número 1 é onde o ficará o servidor central (a) e, junto a ele, um módulo Arduino (b) para monitorar este ambiente. As salas 2 e 3 representam dois ambientes distintos que irão ser monitorados pelo sistema, com acesso à rede por um *switch* (c). A sala 4 representa a casa ou qualquer outro lugar que o usuário esteja,

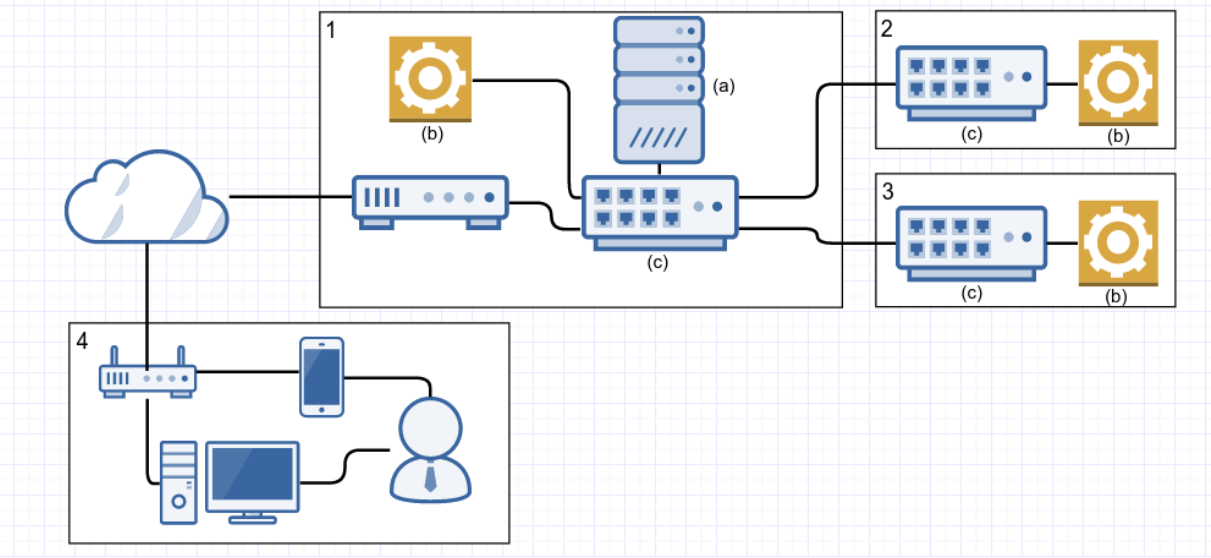

Figura 14 – Arquitetura do sistema

Fonte: O autor

que, via Internet, poderá acessar interface Web do projeto.

4.3.1 SERVIDOR CENTRAL

As obrigações do servidor central podem ser descritas por:

1. Fazer conexão com todos os módulos de forma que seja capaz de receber dados de cada um;
2. Prover uma interface Web para os usuários do sistema em que seja possível realizar configurações do sistema e obter informações sobre os ambientes monitorados;
3. Notificar usuários do sistema sobre inconformidades programadas do ambiente (gatilhos) através de e-mail ou Telegram;
4. Permitir a criação de gatilhos de notificação. Um exemplo de gatilho seria o envio de um e-mail para todos os usuários em caso da temperatura ultrapassar 25°C ou em caso da luminosidade ultrapassar um limite imposto para o ambiente;
5. Guardar dados provenientes dos módulos Arduino.

O desenvolvimento do servidor foi realizado com as seguintes etapas:

- Projeção do banco de dados para armazenar informações necessárias ao sistema;
- Desenvolvimento do sistema Web utilizando o Laravel PHP *framework*;
- Implementação da comunicação entre cliente e servidor utilizando o protocolo MQTT;
- Criação do sistema de notificações baseado nas configurações dos usuários e dados providos dos módulos Arduino.
- Implementação da funcionalidade de gerar o código do módulo pré-configurado.

4.3.1.1 MODELO FÍSICO DO BANCO DE DADOS

A Figura 15 apresenta o modelo físico do banco de dados criado para atender os objetivos do projeto (Seção 1.1).

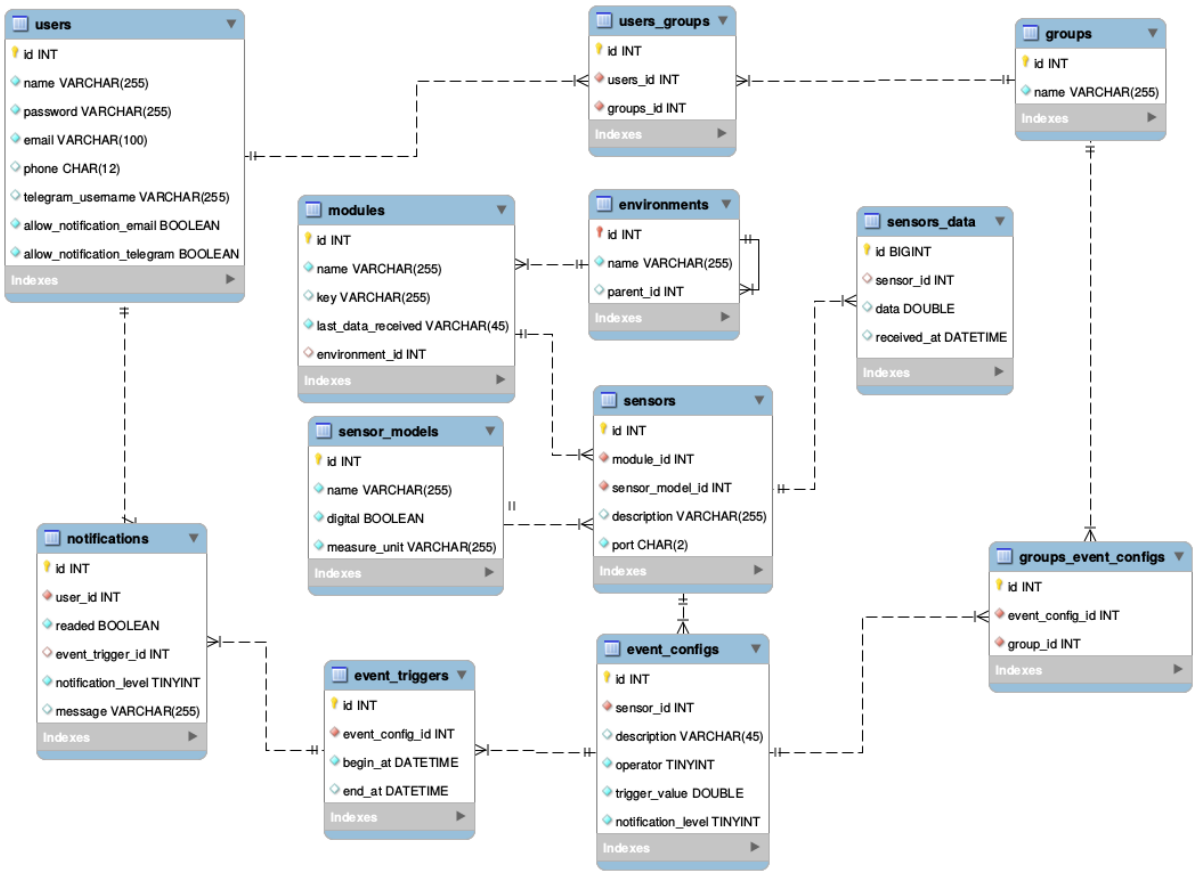

Figura 15 – Modelo físico do banco de dados

Fonte: O autor.

Os dados são organizados de forma que os usuários (tabela *users*) podem estar presentes em vários grupos (tabela *groups*) e possuírem várias notificações (tabela *notifications*).

Os módulos (tabela *modules*) devem pertencer a um ambiente (*environments*) e terem sensores associados a ele (tabela *sensors*).

Os dados provindos dos módulos são diretamente associados à sensores e ficam salvos na tabela *sensors_data*.

O sistema de notificações é constituído por três fundamentais tabelas onde a primeira, *event_configs*, é responsável por definir os gatilhos de disparo de notificações em função dos dados recebidos. A segunda tabela, *event_triggers*, é responsável por controlar o envio das notificações geradas e, a terceira, *notifications*, é utilizada para armazenar as notificações do sistema.

4.3.2 MÓDULO COM ARDUINO

As atribuições do módulo Arduino são:

1. Obter dados do ambiente em que está contido;
2. Ter conexão com o servidor central;
3. Enviar dados recolhidos para o servidor central.

O desenvolvimento do módulo foi constituído por:

- Implementar a comunicação com o servidor;
- Programar a captura das informações do ambiente e envio delas para o servidor.
- Montagem do circuito com o Arduino, *shield* Ethernet e sensores;

OBS: Os sensores não possuirão um circuito padrão. Este varia de acordo com as configurações pré-definidas do módulo no sistema.

5 SOLUÇÃO DESENVOLVIDA

Neste capítulo será apresentada a solução desenvolvida, o sistema 3A - Arduino Ambient Auditor. O objetivo principal do sistema é monitorar ambientes pré-configurados de forma que usuários ou grupos de usuários sejam alertados em caso de anomalias nos dados dos sensores.

5.1 DESCRIÇÃO DA SOLUÇÃO

A solução foi desenvolvida com a finalidade de monitorar vários ambientes instantaneamente por sensores e notificar usuários sobre anomalias em cada um deles.

A solução é constituída por duas principais partes:

- Módulos Arduino: Responsável por capturar dados do ambiente em que está instalado.
- Sistema Web: Responsável por agregar os dados vindos dos módulos, fornecer uma interface para o usuário e enviar notificações em casos de anomalias pré-configuradas.

Cada ambiente a ser monitorado deve possuir um módulo Arduino com sensores instalados e configurados em seu código fonte. A comunicação entre os módulos e o sistema Web é feita através do protocolo de comunicação MQTT que é caracterizado por trabalhar bem em redes lentas ou congestionadas e prover níveis de QoS.

O código-fonte base para os módulos são gerados pelo próprio sistema Web, tornando necessário para os implantadores apenas a programação do filtro dos dados provindos dos sensores.

Para cada sensor configurado, pode-se criar gatilhos de notificações para que o sistema envie-as. As notificações podem ser entregues de três formas: E-mail, Telegram e pelo próprio sistema ao qual possui um histórico de todas as notificações entregues.

Nas próximas sessões será explicada cada parte do sistema e como configurá-lo de acordo com as necessidades impostas.

5.2 CONCEITOS PRINCIPAIS

Nesta seção serão apresentados os conceitos necessários para o entendimento da dinâmica de funcionamento do sistema.

- **Ambientes:** Dentro da solução, cada local a ser monitorado é chamado de ambiente. Ambientes podem possuir um ambiente pai e vários ambientes filhos. Exemplo: O ambiente 1 é um ambiente raiz ao qual representa o câmpus de Guarapuava da UTFPR e, dentro dele, existe um ambiente 2 representando CPD local.
- **Módulos:** Pode-se chamar de módulo um Arduino ao qual possui vários sensores conectados e que seja capaz de enviar dados destes sensores para dentro do sistema Web. Um módulo pode possuir vários sensores, sendo o único limite a versão utilizada do Arduino.

- **Sensores:** Os sensores são as partes responsáveis por captarem dados do ambiente em que estão presentes. Os sensores devem estar conectados ao módulo Arduino e serem configurados diretamente no código-fonte do módulo.
- **Configurações de notificação:** Esta é a parte do sistema em que configura-se os gatilhos para alertar os usuários de acordo com os dados recebidos dos sensores. Podem ser criadas várias configurações de notificações para cada sensor onde podem ter três diferentes níveis de notificação: notificação normal, notificação de alerta e notificação de perigo.

5.3 DINÂMICA DE FUNCIONAMENTO

Nesta seção será explicado, parte por parte, como utilizar e configurar o sistema de acordo com as necessidades impostas.

5.3.1 USUÁRIOS

Inicialmente, o sistema conta com um usuário padrão ao qual recomenda-se usar apenas para cadastrar os outros usuários do sistema. As credenciais do usuário padrão são:

E-mail: admin@admin.com

Senha: admin

Após efetuar o primeiro login é de notória importância alterar a senha do usuário padrão por questões de segurança. O cadastro de novos usuários conta com um formulário onde é definido suas credenciais e as opções para notificações, podendo elas serem enviadas através de e-mail e/ou Telegram.

5.3.2 GRUPOS

As notificações são enviadas tomando como base os grupos de usuários para que, desta forma, possa ser dividida a entrega apenas para as pessoas responsáveis por cada módulo instalado. Exemplo: O sistema está implantado em um câmpus onde há vários blocos. Pode-se definir grupos de pessoas responsáveis por cada bloco a fim de verificar possíveis incidentes.

O sistema também conta com um grupo padrão onde reúne todos os usuários que não estão atribuídos a nenhum outro grupo.

5.3.3 AMBIENTES

A sessão de ambientes foi feita com a finalidade de organizar os módulos. A organização dos ambientes é feita através de hierarquias onde um ambiente pode pode possuir outros ambientes ou pertencer a um ambiente pai. Para o funcionamento do sistema é necessário ter ao menos um ambiente definido para que seja usado como raiz.

5.3.4 MÓDULOS

A sessão de módulos pode ser considerada como uma das partes principais do sistema. É através dos módulos que serão enviadas as informações do ambiente para o sistema, além de agregar os sensores instalados e filtrar os dados obtidos por eles.

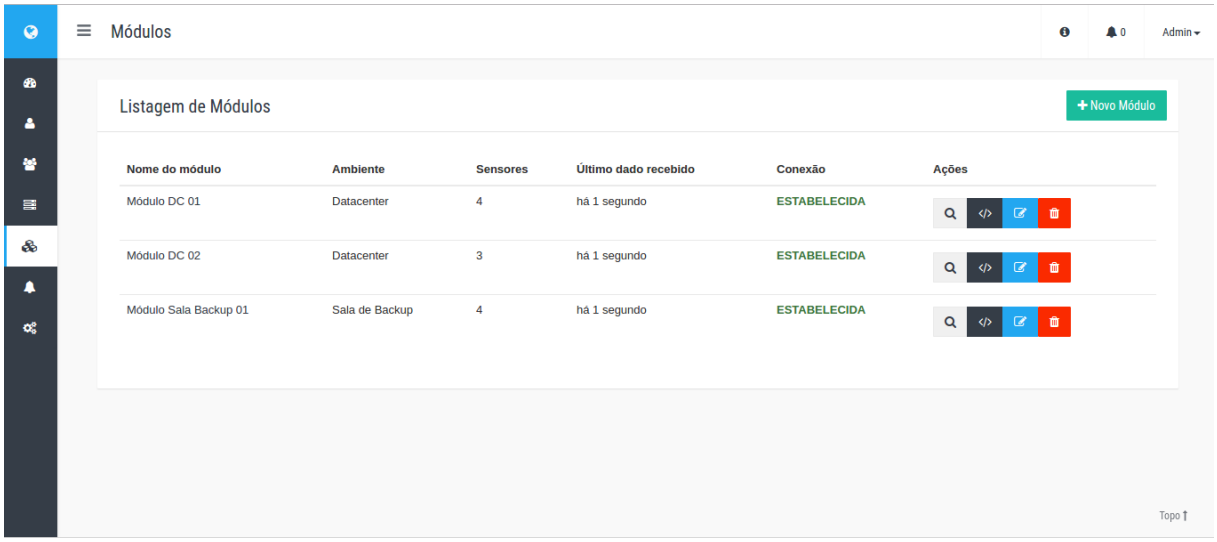

Figura 16 – Listagem de módulos

Fonte: O autor

O cadastro de módulo é relativamente simples contando apenas com o nome e o ambiente ao qual pertence.

Após criado, o módulo já pode ser visto na listagem de módulos, como mostra a Figura 16. Neste momento, o módulo gera automaticamente uma chave de vínculo para que seja possível uma comunicação eficiente com o sistema Web.

A Figura 17 mostra a tela em que é apresentado um módulo recém-cadastrado onde não possui nenhum sensor atribuído. Esta tela é constituída por informações do módulo, botões de ação relacionados diretamente a ele (gerar código base do Arduino, editar e apagar) e botões de ação relacionados aos sensores a serem atribuídos (adicionar sensor).

Observações:

1. É impossível criar um módulo sem atribuir um ambiente;
2. Caso o ambiente em que o módulo esteja atribuído seja apagado, o módulo será desvinculado e não apagado;
3. A chave de vínculo é composta pelos cinco primeiro caracteres de uma *hash* SHA1 feita tomando como base informações como data de criação e nome do módulo. O sistema garante que não sejam criados módulos com *hashs* iguais, repetindo o algoritmo até que o valor seja único.
4. Todas as informações apresentadas na Figura 17 só serão atualizadas após o recarregamento da página.

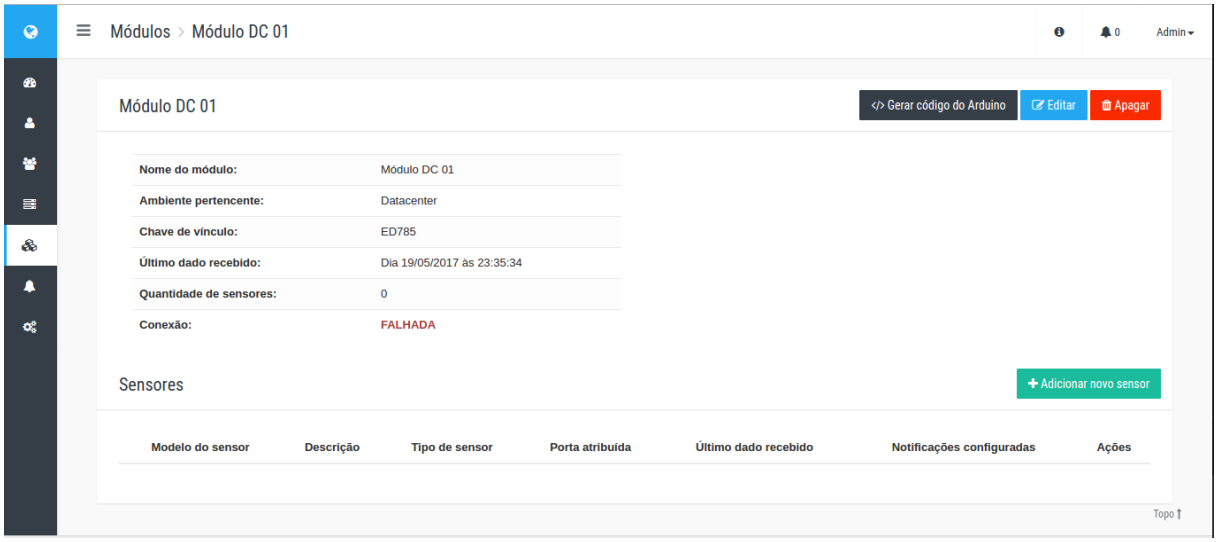

Figura 17 – Exemplo de módulo

Fonte: O autor

5.3.4.1 CONFIGURAÇÃO DOS SENSORES

Para cada sensor que será conectado fisicamente na placa, obrigatoriamente deverá estar configurado no sistema Web, e vice-versa, caso contrário, o sistema não conseguirá obter os dados referentes aos sensores e, consequentemente, não realizará a tarefa de notificar os usuários em casos de anomalias.

O formulário para a configuração de um sensor é composto por dois campos fundamentais:

- **Modelo do sensor:** Define qual modelo de sensor será utilizado. Modelos de sensores podem ser cadastrados através da interface Web do sistema.
- **Porta:** Define em qual porta o sensor será conectado.

Observações:

1. Não é possível atribuir dois sensores na mesma porta.
2. O campo "Porta" pode receber valores entre 0-99 (portas digitais) ou A1-A9 (portas analógicas).
3. Sensores analógicos conectados em portas digitais automaticamente são reconhecidos como PWM (*Pulse Width Modulation*). O sistema não verifica se a porta realmente funciona como PWM, sendo esta uma restrição variável de acordo com o modelo do Arduino utilizado.
4. A precisão do sistema é diretamente relacionada à precisão dos sensores utilizados.

5.3.4.2 GERANDO O CÓDIGO FONTE DE BASE DO MÓDULO

A solução desenvolvida também gera o código base para o módulo Arduino através das configurações do módulo e dos sensores configurados.

O código gerado não realiza nenhum tratamento nos dados dos sensores. Para fazer os tratamentos, o código gerado deve ser modificado por pessoas com, ao menos, conhecimento básico sobre Arduino e programação.

Para obter o código base, deve-se clicar no botão "Gerar código do Arduino" (apresentado na Figura 17 junto aos botões de ação relacionados diretamente ao módulo). Ao ser clicado no botão, o usuário será redirecionado à uma tela de configuração apresentada na Figura 18 abaixo.

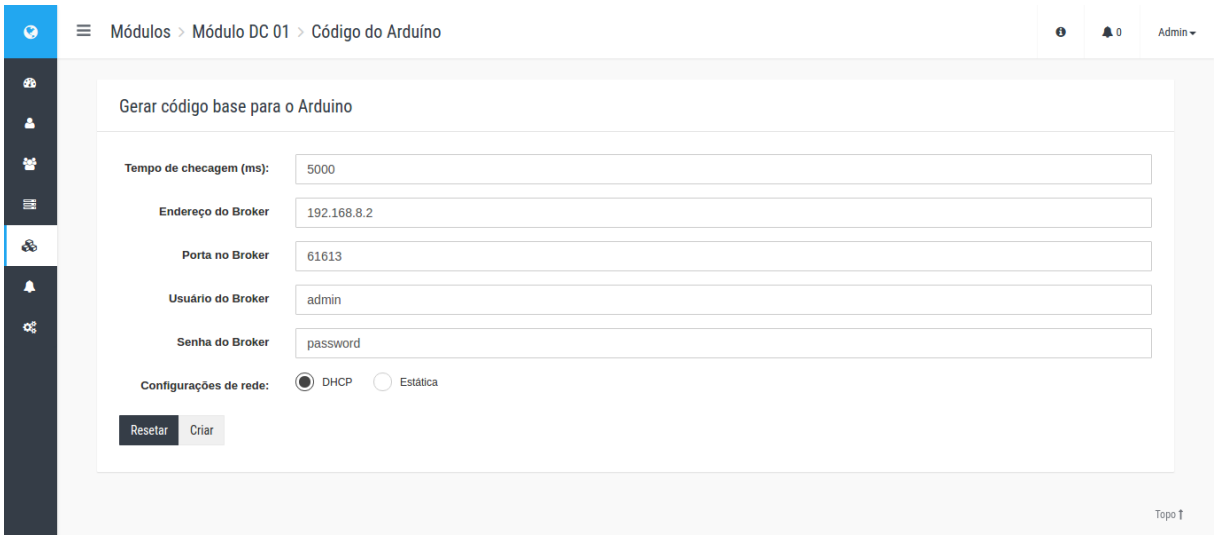

Figura 18 – Formulário para gerar código base para o Arduino

Fonte: O autor

O formulário de configuração para gerar o código-fonte base para o Arduino (Figura 18) possui inicialmente seis campos obrigatórios. O primeiro trata-se do tempo (em milissegundos) de intervalo para enviar os dados dos sensores para o sistema Web. No segundo e terceiro campo deve ser inserido o IP e a porta, respectivamente, do *broker* utilizado (Apache Apollo). O quarto e o quinto campo devem ser preenchidos com as credenciais do usuário configurado no *broker*. O último campo trata-se da forma em que o Módulo será configurado na rede. A opção "DHCP" fará com que o serviço de DHCP (*Dynamic Host Configuration Protocol*), presente na rede, provenha as configurações automaticamente. A opção "Estática" forçará o usuário a configurar o IP do módulo.

Após o preenchimento do formulário, deve-se acionar o botão "Criar" para que o sistema gere o código fonte base do módulo Arduino. A Figura 19 contém o código gerado e a opção de baixá-lo como *sketch* para abri-lo diretamente na IDE do Arduino.

Figura 19 – Tela com o código gerado

Fonte: O autor

Observações:

1. As regras de validação para os campos relacionados à configuração estática para rede foram criadas de acordo com as restrições impostas pelo Arduino.
2. Após passar o código compilado para o módulo Arduino e ligá-lo à rede, a informação "Conexão" apresentada na Figura 17 deve mostrar seu valor como "ESTABELECIDA" ao invés de "FALHADA".
3. O formulário de configurações para gerar o código base do módulo traz como valores padrão as configurações definidas na sessão de "configuração" do sistema.
4. Para realizar a compilação do código Arduino é necessário ter instalado a biblioteca "PubSubClient", disponível no endereço vinculado com o *hyperlink* "PubSubClient no Arduino" apresentado na Figura 19.
5. O endereço MAC (*Media Access Control*) deve ser configurado manualmente para cada módulo gerado.

5.3.5 NOTIFICAÇÕES

As notificações são dadas como o recurso chave da solução desenvolvida. Nesta sub-seção será apresentado como configurá-las e como vê-las dentro do sistema.

5.3.5.1 CONFIGURANDO NOTIFICAÇÕES PARA OS SENSORES

Uma configuração de notificação também pode ser chamada de gatilho de notificação que é onde configura-se as ocasiões em que o sistema irá disparar os avisos. Existem dois caminhos para configurar o envio de notificações. O primeiro é clicando no botão para criar novo evento de configuração na listagem das configurações de notificações. A segunda forma

é clicando no botão para adicionar novo gatilho de notificação diretamente na listagem de sensores de um módulo.

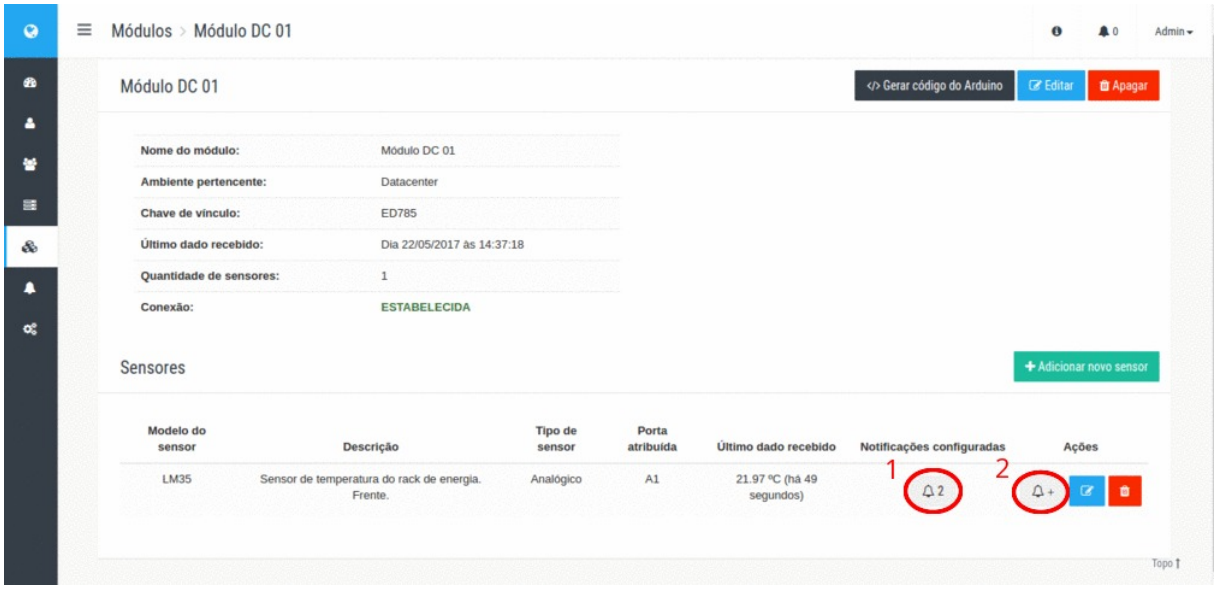

Figura 20 – Tela de visualização do módulo e listagem dos sensores atribuídos.

Fonte: O autor

A Figura 20 apresenta um módulo com apenas um sensor configurado. A indicação em vermelho marcada pelo número 1 representa o botão para listar os gatilhos de notificações criados, enquanto o número 2 representa o botão para criar um novo gatilho de notificação. Cada gatilho de notificação funciona apenas para um único sensor. Podem ser criados vários gatilhos para um único sensor.

Um gatilho é composto por cinco características fundamentais:

1. **Sensor:** Sensor ao qual será atribuído o gatilho.
2. **Operador:** Operador para comparação entre o dado recebido e o valor de disparo.
3. **Valor de disparo:** Valor ao qual será usado para verificar se a notificação será enviada ou não de acordo com o operador definido e o dado recebido.
4. **Nível de notificação:** Pode ser usado 3 valores neste campo: normal, alerta ou perigo. Usado para criar níveis de notifição.
5. **Grupos a serem notificados:** Devem ser escolhidos todos os grupos responsáveis pelo ambiente. A notificação será emitida, individualmente, para cada membro dos grupos.

Um gatilho deve ser interpretado da seguinte forma:

O dado recebido no <**sensor**> é <**operador**> do que <**valor de disparo**>. Caso positivo, envie notificações do tipo <**nível de notificação**> para <**grupos configurados**>. Exemplo:

O dado recebido em **Datacenter -> Módulo DC 01 -> Sensor de temperatura do rack de energia. Frente. (LM35 na porta A1)** é **>= (maior ou igual)** do que **30ºC**. Caso positivo, envie notificações do tipo "**alerta**" para **responsáveis pelo bloco A**.

Para cada incidente o sistema envia duas notificações, uma quando detecta o incidente e outra, com a mesma prioridade, quando o sensor volta a atingir os valores normais para o gatilho.

5.3.5.2 LISTAGEM E RECONHECIMENTO DAS NOTIFICAÇÕES RECEBIDAS

Além de enviar as notificações por e-mail e por Telegram, o sistema também possui uma central de notificações onde ficam registradas todas as notificações emitidas o usuário corrente em que ele está vinculado.

As notificações possuem dois estados: reconhecidas e não reconhecidas. Uma notificação é dada como reconhecida quando o usuário está ciente de que ela foi emitida e a marca como reconhecida dentro do sistema. A barra de navegação superior do sistema (Figura 21) mostra a quantidade de notificações ainda não reconhecidas o usuário possui.

Figura 21 – Barra de navegação superior.

Fonte: O autor

Ao clicar no botão representado por um sino, aparecerá o total de notificações não reconhecidas separadas pelos níveis de notificação, e, logo em seguida, um botão que redireciona o usuário para a listagem de todas as notificações emitidas para ele.

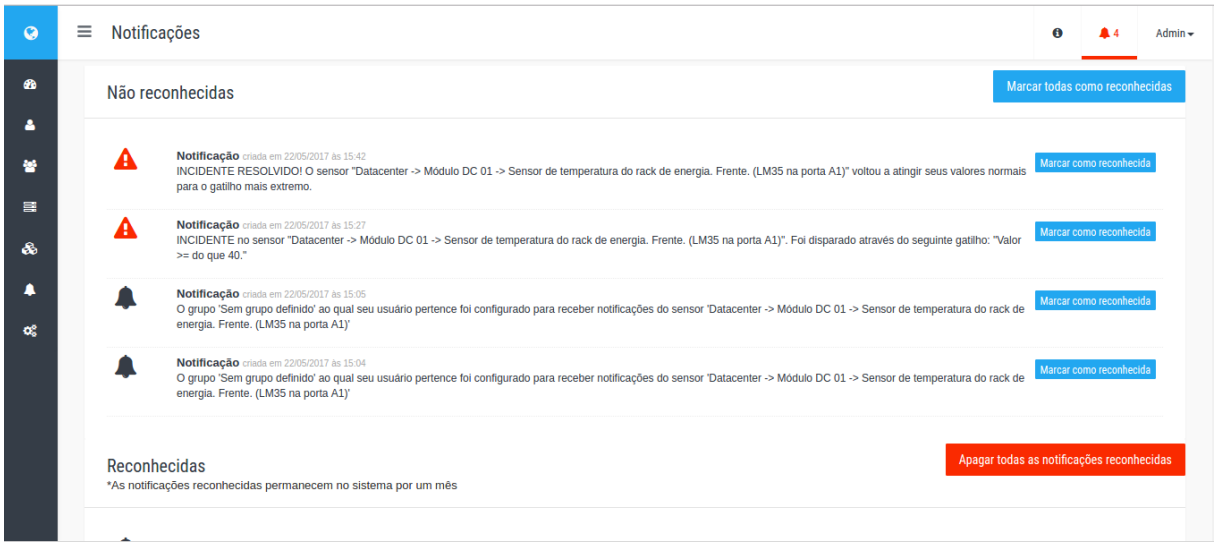

Figura 22 – Tela que apresenta a listagem de notificações emitidas para o usuário corrente.

Fonte: O autor

A Figura 22 mostra a tela em que lista-se todas as notificações emitidas para o usuário corrente no sistema. Existe uma separação entre as notificações reconhecidas e as notificações não reconhecidas.

Apenas as notificações já reconhecidas podem ser apagadas. Estas permanecem no sistema por, no máximo, um mês. Após este tempo serão apagadas automaticamente.

5.3.6 DASHBOARD E GRÁFICOS

É chamada de *dashboard* a tela em que agrega gráficos e informações importantes sobre os ambientes monitorados.

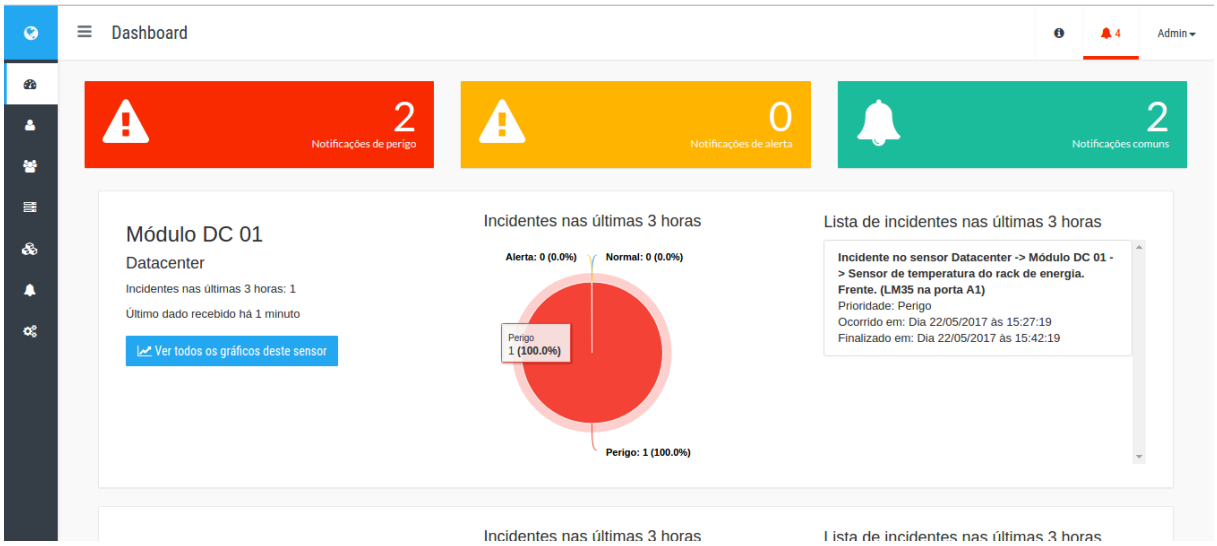

Figura 23 – Dashboard do sistema.

Fonte: O autor

A *dashboard*, apresentada na Figura 23, é a tela inicial do sistema onde contém *cards* com a quantidade de notificações recebidas separadas por nível de alerta. Também constitui esta tela um breve resumo dos incidentes ocorridos em cada módulo, contendo gráficos e a listagem de cada um deles. Além dos dados de incidentes, é apresentada também a data do último dado recebido e um botão que redireciona o usuário para ver todos os gráficos do módulo em questão.

Para esta versão inicial da solução não é possível definir o intervalo de busca de informações na *dashboard*, sendo este fixado em 3 horas.

Para conseguir ver um histórico maior de dados, é necessário abrir a tela de gráficos clicando no botão "Ver todos os gráficos deste módulo", localizado em cada módulo mostrado na *dashboard*.

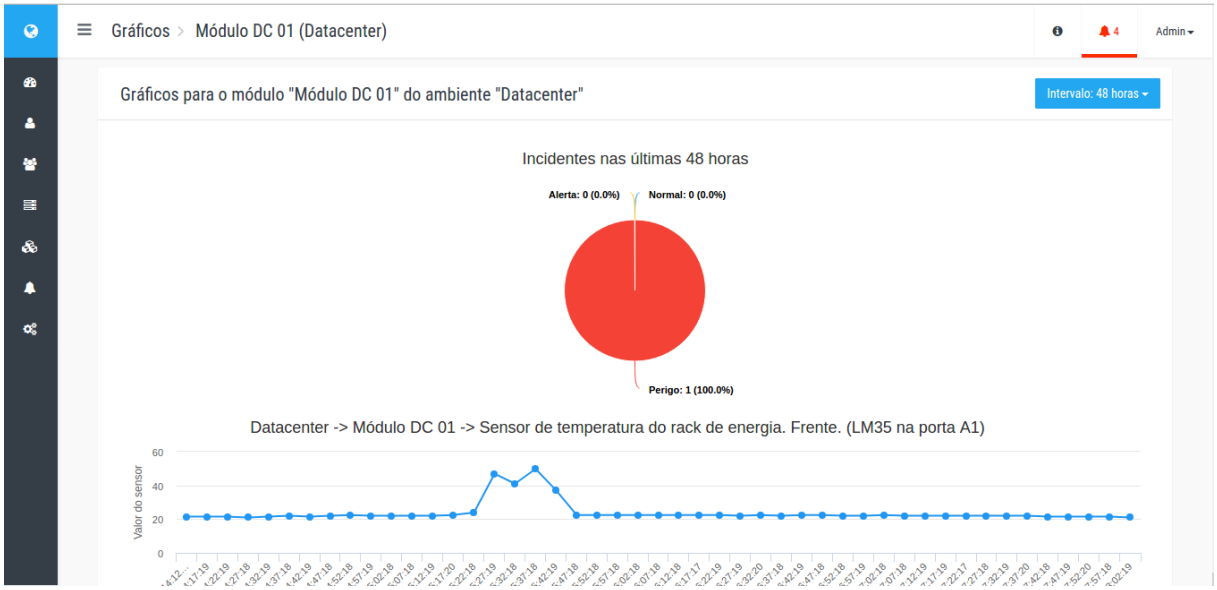

Figura 24 – Gráficos gerados para um determinado módulo.

Fonte: O autor

A Figura 24 apresenta a tela de gráficos gerados para um determinado módulo (“Módulo DC 01” do ambiente “Datacenter” neste caso). Do lado direito da tela, logo abaixo da barra de navegação superior (Figura 21) existe o botão para selecionar o intervalo de consulta, podendo ser ele de 2 horas, 3 horas, 5 horas, 8 horas, 12 horas, 1 dia ou 2 dias.

6 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo desenvolver um sistema para monitoramento ambiente utilizando a plataforma Arduino juntamente com uma interface Web para visualização e notificação de anomalias pré-configuradas.

No decorrer do desenvolvimento do sistema, foram feitas breves reuniões com a equipe de TI (Tecnologia da Informação) da UTFPR câmpus Guarapuava sobre possíveis ajustes e modificações para facilitar a utilização da interface gráfica.

A solução desenvolvida no decorrer deste trabalho foi feita utilizando componentes de baixo custo e tecnologias *open-souce* possibilitando que a sua implantação possa ser feita sem alto investimento.

Nas seções abaixo, apresenta-se a a contribuição e possibilidades de trabalhos futuros sobre a solução desenvolvida.

6.1 CONTRIBUIÇÃO DO TRABALHO

Atualmente o sistema encontra-se implantado no câmpus Guarapuava da UTFPR devido ao fato de dar suporte e fornecer o ambiente para testes durante o desenvolvimento.

Pelo fato da UTFPR prover todo recurso necessário para aplicação e implantação do trabalho, a solução apresentada poderá ser utilizada em qualquer CPD dentro da instituição de forma gratuita.

6.2 TRABALHOS FUTUROS

Esta seção tem como finalidade apresentar melhorias futuras para a solução desenvolvida.

1. **Implementação de ACLs (Access Control List):** A solução apresentada neste trabalho não possui nenhuma forma de controle de acesso aos recursos oferecidos, sendo assim, todos os usuários cadastrados possuem o nível de administrador do sistema. Uma futura atualização do projeto poderia conter este recurso.
2. **Implementação de outros protocolos de comunicação:** A solução permite a troca do protocolo de comunicação entre o servidor WEB e o módulo Arduino. Uma possível continuação do trabalho poderia trazer implementações de protocolos de comunicação variados a fim de realizar testes em redes ou do próprio protocolo.
3. **Realizar ações no sistema através da interação com o BOT no Telegram:** Na versão apresentada, a utilização do BOT no Telegram se dá em apenas receber notificações. Uma atualização futura poderia permitir enviar comandos para o servidor afim de realizar ações pré-programadas, como gerar relatórios e enviar por PDF ou para enviar um comando para ativar um atuador em algum dos módulos Arduino.

Referências

ACTIVEMQ. **Apollo: ActiveMQ's next generation of messaging**. 2016. Acesso em: 10 mai. 2016. Disponível em: <https://activemq.apache.org/apollo/>. Citado na página 21.

ADRIEL, W. **Introdução ao Framework PHP Laravel**. 2015. Acesso em: 19 mai. 2016. Disponível em: <http://www.devmedia.com.br/introducao-ao-framework-php-laravel/33173>. Citado 2 vezes nas páginas 4 e 5.

AGRIYA. **Pros and Cons of most Favoured PHP Frameworks**. 2015. Acesso em: 07 jun. 2016. Disponível em: <https://www.agriya.com/blog/2015/08/31/pros-cons-favoured-php-frameworks>. Citado na página 5.

ALIEXPRESS. **13 em 1 Starter Kit Módulos de Sensores de Chama para Arduino + MCU Uso Educacional**. 2016. Acesso em: 04 jun. 2016. Disponível em: <http://pt.aliexpress.com/item/13-in-1-Sensors-Modules-Starter-Kit-Flame-for-Arduino-MCU-Education-User-TE272/32572750949.html?spm=2114.42010708.4.4.uRa5cv>. Nenhuma citação no texto.

ALVAREZ, M. A. **Breve história do PHP**. 2004. Acesso em: 19 mai. 2016. Citado na página 4.

ARDUINO. **Arduino Ethernet Shield**. 2016. Acesso em: 07 abr. 2016. Disponível em: <https://www.arduino.cc/en/Main/ArduinoEthernetShield>. Citado na página 9.

ARDUINO. **Arduino Mega**. 2016. Acesso em: 07 abr. 2016. Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardMega>. Citado na página 7.

ARDUINO. **Arduino Shields**. 2016. Acesso em: 07 abr. 2016. Disponível em: <https://www.arduino.cc/en/Main/ArduinoShields>. Citado na página 8.

ARDUINO. **Arduino Software**. 2016. Disponível em: <https://www.arduino.cc/en/Main/Software>. Citado na página 20.

ARDUINO. **Arduino UNO & Genuino UNO**. 2016. Acesso em: 07 abr. 2016. Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardUno>. Citado na página 7.

ARDUINO. **Arduino Yún**. 2016. Acesso em: 07 abr. 2016. Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardYun>. Citado na página 7.

ARDUINO. **Introduction: What is Arduino?** 2016. Acesso em: 07 abr. 2016. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Citado na página 7.

ARDUINO. **Referência da Linguagem**. 2016. Disponível em: <http://playground.arduino.cc/Portugues/Referencia>. Citado na página 20.

ARDUINOECIA. **Sensor de umidade e temperatura DHT11**. 2013. Acesso em: 03 jun. 2016. Disponível em: <http://www.arduinoecia.com.br/2013/05/sensor-de-umidade-e-temperatura-dht11.html>. Citado na página 12.

ASSOCIATION, T. I. techreport, **Telecommunications Infrastructure Standard for Data Centers**. 2005. Disponível em: <https://manuais.iessanclemente.net/images/9/9f/Tia942.pdf>. Nenhuma citação no texto.

BANAVAR, G. et al. An efficient multicast protocol for content-based publish-subscribe systems. In: **Distributed Computing Systems, 1999. Proceedings. 19th IEEE International Conference on**. [S.l.: s.n.], 1999. p. 262–272. ISSN 1063-6927. Citado na página 14.

BRITTO, R. A. R. C. C. P. J. A. **Contador de Passagens**. 2003. Acesso em: 08 abr. 2016. Disponível em: <http://www.gta.ufrj.br/grad/01_1/contador555/contador_de_passagem.htm>. Citado na página 10.

BROWN, K. **Desenvolva um sensor de temperatura pronto para nuvem com o Arduino Uno e o IBM IoT Foundation, Parte 2**. 2014. Acesso em: 20 mai. 2016. Disponível em: <http://www.ibm.com/developerworks/br/cloud/library/cl-bluemix-arduino-iot2>. Citado na página 13.

CUNHA, A. F. **O que são sistemas embarcados**. [S.l.]: Saber Eletrônica, 2007. v. 43. Nenhuma citação no texto.

ELETRICAL4U. **Light Dependent Resistor | LDR and Working Principle of LDR**. 2016. Acesso em: 28 mai. 2016. Disponível em: <http://www.electrical4u.com/light-dependent-resistor-ldr-working-principle-of-ldr/>. Citado na página 11.

EUROTECH, I. B. M. C. I. **MQTT V3.1 Protocol Specification**. [S.l.], 2010. Acesso em: 25 mai. 2016. Disponível em: <http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html>. Citado na página 13.

EVANS, D. The internet of things: How the next evolution of the internet is changing everything. **Cisco Internet Business Solutions Group**, p. 11, 2011. Acesso em: 30 mar. 2016. Disponível em: <http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf>. Nenhuma citação no texto.

FILIPEFLOP. **COMUNICAÇÃO PELA REDE COM O MÓDULO ETHERNET ENC28J60**. 2016. Acesso em: 04 jun. 2016. Disponível em: <http://blog.filipeflop.com/modulos/modulo-ethernet-enc28j60-arduino.html>. Nenhuma citação no texto.

FUCTURA. **Vantagens e desvantagens do PHP**. 2016. Acesso em: 25 mai. 2016. Disponível em: <http://www.fuctura.com.br/2013/01/vantagens-e-desvantagens-do-php/>. Citado na página 4.

GILMORE, W. J. **Dominando PHP e MySQL: Do iniciante ao profissional**. Rio de Janeiro: Alta Books editora, 2008. Nenhuma citação no texto.

HUNKELER, U.; TRUONG, H. L.; STANFORD-CLARK, A. Mqtt-s; a publish/subscribe protocol for wireless sensor networks. In: **Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on**. [S.l.: s.n.], 2008. p. 791–798. Citado na página 13.

KANNAN, S. **Paho Python client for MQTT and G-code Visualization Talks, Chennaipy**. 2015. Acesso em: 04 jun. 2016. Disponível em: <http://www.shakthimaan.com/posts/2015/10/13/chennai-py-talks/news.html>. Nenhuma citação no texto.

KITRONIK. **How an LDR works**. 2014. Disponível em: <https://www.kitronik.co.uk/pdf/How_an_LDR_works.pdf>. Citado na página 11.

LEE, S. et al. Correlation analysis of mqtt loss and delay according to qos level. In: **The International Conference on Information Networking 2013 (ICOIN)**. [S.l.: s.n.], 2013. p. 714–717. ISSN 1550-445X. Citado na página 13.

LIMA, G. F. Controle de temperatura de um sistema de baixo custo utilizando a placa arduino. In: **IX Congresso de iniciação científica do IFRN**. [s.n.], 2013. Acesso em: 01 mai. 2016. Disponível em: <http://www2.ifrn.edu.br/ocs/index.php/congic/ix/paper/viewFile/765/134>. Citado na página 8.

MCROBERTS, M. **Arduino Básico**. [S.l.]: Novatec Editora, 2010. Nenhuma citação no texto.

MILANI, A. **MySQL - Guia do Programador**. [S.l.]: Novatec Editora, 2007. ISBN 9788575221037. Citado na página 6.

MODTRONIX. **DHT11 Temperature and Humidity Sensor Module**. 2016. Acesso em: 04 jun. 2016. Disponível em: <http://modtronix.com/mod-dht11.html>. Nenhuma citação no texto.

MUKHOPADHYAY, S. C. **Internet of Things: Challenges and Opportunities**. 2014. ed. [S.l.]: Springer, 2014. (Smart Sensors, Measurement and Instrumentation, 9). Citado na página 1.

MYSQL. **MySQL: The world's most popular open source database**. 2016. Acesso em: 27 mai. 2016. Disponível em: <mysql.com>. Citado na página 6.

NATIONALSEMICONDUCTOR. **LM35: Precision Centigrade Temperature Sensors**. [S.l.], 2000. Acesso em: 28 mai. 2016. Disponível em: <http://www.webtronico.com/documentos/LM35.pdf>. Citado na página 11.

NINJAGECKO. **MEASURING LIGHT USING YOUR PI**. 2016. Acesso em: 03 jun. 2016. Disponível em: <http://www.ninjagecko.co.uk/how-to-measure-the-amount-of-light-in-a-room-using-a-light-dependent-resistor/>. Nenhuma citação no texto.

OPENSOURCE. **What is open source ?** 2016. Acesso em: 07 abr. 2016. Disponível em: <https://opensource.com/resources/what-open-source>. Citado na página 7.

PACIEVITCH, Y. **HTML**. 2016. Acesso em: 27 mai. 2016. Disponível em: <http://www.infoescola.com/informatica/html/>. Citado na página 3.

PARSKO, L. F. **TUTORIAL Aplicações, Funcionamento e Utilização de Sensores**. 2006. Acesso em: 27 mai. 2016. Disponível em: <http://www.maxwellbohr.com.br/downloads/robotica/mec1000_kdr5000/tutorial_eletronica_-_aplicacoes_e_funcionamento_de_sensores.pdf>. Citado na página 10.

REYES, J. **15 Wonderfully Creative Uses for PHP**. 2009. Acesso em: 20 mai. 2016. Disponível em: <http://code.tutsplus.com/tutorials/15-wonderfully-creative-uses-for-php--net-4714>. Nenhuma citação no texto.

SADDAM. **Arduino Based Digital Thermometer**. 2015. Acesso em: 05 jun. 2016. Disponível em: <http://circuitdigest.com/microcontroller-projects/digital-thermometer-using-arduino>. Nenhuma citação no texto.

SCHIMIGUEL, P. T. B. J. Desenvolvimento de aplicações mobile cross-platform utilizando phonegap. **Revista Observatorio da Economia Latinoamericana**, 2016. Disponível em: <http://www.eumed.net/cursecon/ecolat/br/16/phonegap.html>. Citado na página 4.

SHELDON, A. O. R. **SQL: Um guia para iniciantes**. 3. ed. Rio de Janeiro: Editora ciência moderna Ltda., 2009. Citado na página 5.

SILVA, M. S. **Introdução às CSS**. 2003. 10/12/2003. Acesso em: 28 mai. 2016. Disponível em: <http://www.maujor.com/tutorial/intrtut.php>. Citado na página 3.

SOARES, W. **PHP5: Conceitos, Programação e Integração com Banco de Dados**. 6. ed. São Paulo: Editora Érica Ltda., 2010. Citado na página 4.

STEFFENS, C. A. **O funcionamento e uso de alguns sensores usando a placa de som de um PC**. 2016. Aquisição de dados em laboratório didático de física. Acesso em: 08 abr. 2016. Disponível em: <http://www.if.ufrgs.br/mpef/mef004/20061/Cesar/SENSORES-Definicao.html>. Citado na página 9.

SUNROM. **DHT11: Humidity and Temperature Sensor**. [S.l.], 2012. Acesso em: 02 jun. 2016. Disponível em: <http://robocraft.ru/files/datasheet/DHT11.pdf>. Citado na página 12.

TELEGRAM. **Bots: An introduction for developers**. 2016. Acesso em: 03 jun. 2016. Disponível em: <https://core.telegram.org/bots>. Nenhuma citação no texto.

TELEGRAM. **Telegram FAQ**. 2016. Acesso em: 03 jun. 2016. Disponível em: <https://telegram.org/faq>. Citado na página 14.

TIOBE. **TIOBE Index for June 2016**. 2016. Acesso em: 01 jun. 2016. Disponível em: <http://www.tiobe.com/tiobe_index?page=index>. Nenhuma citação no texto.

TUTORIALSPOINT. **SQL TUTORIAL**. 2016. Acesso em: 03 jun. 2016. Disponível em: <http://www.tutorialspoint.com/sql/sql_tutorial.pdf>. Nenhuma citação no texto.

W3SCHOOLS. **CSS Tutorial**. 2016. Acesso em: 01 jun. 2016. Disponível em: <http://www.w3schools.com/css/>. Citado na página 3.

W3SCHOOLS. **HTML5 Introduction**. 2016. Acesso em: 01 jun. 2016. Disponível em: <http://www.w3schools.com/html/html5_intro.asp>. Citado na página 3.

XDSOFTWARE. **Apostila de SQL**. 2016. Acesso em: 20 mai. 2016. Disponível em: <http://www.xdsoftware.com/demo/BR-pt/manuais/ManSQL1.pdf>. Citado na página 5.