UNIVERSIDADE PAULISTA

CIÊNCIA DA COMPUTACÃO

RUTY RIBEIRO (ORG.)

**“DESENVOLVIMENTO DE SISTEMA DE IDENTIFICAÇÃO E AUTENTICAÇÃO BIOMÉTRICA”**

São Paulo

2020

JOÃO PAULO STEFANI FREGNI – N346035

MARLLON DOMINGOS – RA:N314DB2

RAFAEL BISPO DOS SANTOS – RA:D482HF1

RUTY RIBEIRO - RA: D826441

WEZA NARCIA PINTO MINGOSSO – RA: D815IF8

**“DESENVOLVIMENTO DE SISTEMA DE IDENTIFICAÇÃO E AUTENTICAÇÃO BIOMÉTRICA”**

Trabalho apresentado como exigência parcial de nota. Curso: Ciência da Computação. Disciplina: Atividades De Práticas Supervisionadas. Período: noturno. Série: V/VI. Turma: CC4P39/CC5P39. Unidade: Anchieta. Prédio: A. Sala: 313. Universidade Paulista (UNIP)

Orientador: Professor Ricardo Piantola.

São Paulo

2020

# SUMÁRIO

**1. INTRODUCÃO**3

**2. OBJETIVO**5

2.1OBJETIVOS GERAIS5

2.2OBJETIVOS ESPECÍFICO5

**3.** **FUNDAMENTOS DAS PRINCIPAIS TÉCNICAS BIOMÉTRICAS** 6

3.1.XXXXXXXXXXXXXXXXX6

3.1.1. XXXXXXXXXXXXXXXXX6

3.1.1.1.XXXXXXXXXXXXXXXX7

3.1.1.2.XXXXXXXXXXXXXXXX 7

3.1.1.3.XXXXXXXXXXXXXXXX 7

3.1.1.4.XXXXXXXXXXXXXXXX8

3.1.1.5. XXXXXXXXXXXXXXXX 8

3.1.1.6. XXXXXXXXXXXXXXXX 8

3.1.1.7. XXXXXXXXXXXXXXXX 8

3.1.2. XXXXXXXXXXXXXXXX.8

3.1.2.1. XXXXXXXXXXXXXXXX 9

3.1.2.2. XXXXXXXXXXXXXXXX10

3.1.2.3. XXXXXXXXXXXXXXXX 10

3.1.2.4. XXXXXXXXXXXXXXXX DE REDE11

**4.** **PLANO DE DESENVOLVIMENTO DA APLICAÇÃO**12

4.1.XXXXXXXXXXXXXXXX 12

4.2.XXXXXXXXXXXXXXXX 13

4.2.1.XXXXXXXXXXXXXXXX 13

4.2.1.1 XXXXXXXXXXXXXXXX 14

4.3.XXXXXXXXXXXXXXXX 14

4.4.XXXXXXXXXXXXXXXX 16

**5.** **O** **PROJETO**17

5.1. XXXXXXXXXXXXXXXX 17

5.2. XXXXXXXXXXXXXXXX 19

5.3 XXXXXXXXXXXXXXXX 19

**6.** **LINHAS DE CÓDIGO**20

6.1. XXXXXXXXXXXXXXXX 20

6.2. XXXXXXXXXXXXXXXX 21

6.3. XXXXXXXXXXXXXXXX 21

**7.** **APRESENTAÇÃO DO PROGRAMA**22

**REFERÊNCIAS**23

**FICHAS DE ATIVIDADES PRATICAS SUPERVISIONADAS** 41

**1. INTRODUÇÃO**

**2. OBJETIVO**

2.1 OBJETIVO GERAL

O objetivo desse trabalho consiste em implementar um projeto web

2.2 OBJETIVOS ESPECÍFICOS

1 – Fundamentar a comunicação de dados em redes;

2 – Apresentar todo o plano do desenvolvimento do projeto IoT;

3 – Subclassificar e demostrar cada etapa realizada no desdobramento do projeto IoT.

**3. FUNDAMENTOS DA COMUNICAÇÃO DE DADOS EM REDE**

A partir da década de 70 o ramo da tecnologia da informação teve um grande avanço com a ascensão da rede de computadores gerando uma fusão entre a Ciência da Computação e a Comunicação de dados. Por definição a rede é uma maneira de conectar computadores de forma na qual eles compartilhem recursos, sejam eles físicos ou lógicos.

Indubitavelmente a Rede de Computadores revolucionou e facilitou muitos processos, seja em ambientes corporativos ou domésticos pois oferece benefícios como: permitir acesso simultâneo a dados importantes, compartilhamento em dispositivos periféricos, agilizar o processo de backups e etc.

As redes de comunicação exercem uma função extremamente importante no mundo globalizado porque a sua constante evolução permite que informações sejam compartilhadas através do mundo inteiro com rapidez e eficiência.

3.1. A COMUNICAÇÃO DE DADOS

Um sistema básico de comunicação de dados em rede é composto por 5 componentes básicos: a mensagem, o transmissor, o receptor, o meio e o protocolo onde a mensagem, qualquer que seja a sua natureza, será enviada através do meio físico pelo dispositivo transmissor até o receptor enquanto o protocolo exerce uma espécie de mediação entre a comunicação junto dos dois dispositivos.

3.1.1. O MODELO OSI

O modelo OSI (Open System Interconnection) foi criado em 1971 com intuito de padronizar os protocolos de comunicação de dados em rede. Este padrão divide a rede de computadores em nada mais que 7 camadas diferentes, de modo que apenas a camada N+1 poderá referir-se à entidade N.

A partir da figura acima podemos perceber que os níveis são nomeados de baixo para cima, isso significa que quando mais baixo o nível mais perto do hardware a camada estará.

Cada nível nesse sistema domina distintas partições de extrema importância para que os dados sejam devidamente transmitidos:

3.1.1.1. A CAMADA FÍSICA

Esta camada trata-se do nível mais baixo do modelo OSI (Camada 1), é nela onde todo o processo de comunicação é iniciado, é nela onde todas as características mecânicas, elétricas, funcionais e os procedimentos para ativar, manter e desativar conexões físicas estão localizados.

O nível 1 recebe os dados em forma de pulsos elétricos e transforma-os em bits, ou seja, 0 e 1. Nos cabos apenas sinais de baixa frequência são transmitidos, nesta camada são identificados os bits de digito “0” como sinal elétrico de –5 volts, logo os bits de digito “1” como sinal elétrico com +5 volts.

 3.1.1.2. A CAMADA DE ENLACE

A camada de enlace ou ligação de dados trata-se do segundo nível do modelo OSI. Como dito anteriormente, cada camada pode comunicar-se apenas com uma outra, neste caso a de enlace incorpora os bits anteriormente traduzidos pela física converte-os para byte, por exemplo, transformando em unidade de dado, logo subtrai o endereço físico (MAC address) e encaminha os resultados para o nível seguinte.

 3.1.1.3. A CAMADA DE REDE

É nesse nível onde as rotas são definidas, totalmente responsável pelo processo de internetworking, a partir de dispositivos como os roteadores ela é encarregada por definir qual rota seria mais viável no processo de interconexão. É importante citar que essa camada transforma o endereço físico em um endereço lógico (conhecido como IP).

3.1.1.4. A CAMADA DE TRANSPORTE

A camada de enlace ou ligação de dados trata-se do quarto nível do modelo OSI. Essa camada é responsável por nada menos que a entrega/recebimento dos dados encaminhados e endereçados vindos do nível de rede.

A camada de transporte assegura que o tráfego de dados seja seguro e confiável. Nesse grau junto ao IP são anexados os protocolos de transporte.

3.1.1.5. A CAMADA DE SESSÃO

Após todo o caminho percorrido durante as quatro etapas acima a camada de sessão entra em cena iniciando, gerenciando e finalmente terminando a comunicação/troca de dados entre hosts. Ao gerenciar a comunicação o nível de sessão deve estar atento a sincronização de ambas os dispositivos para que a sessão aberta mantenha-se funcionando

3.1.1.6. A CAMADA DE APRESENTAÇÃO

A Camada de Apresentação pode ser definida como o grau de intermediação porque ela basicamente formata e apresenta os dados transmitidos, com isso é possível fazer com que duas redes diferentes se comuniquem. Existem alguns dispositivos que atuam nesse processo como o Gateway.

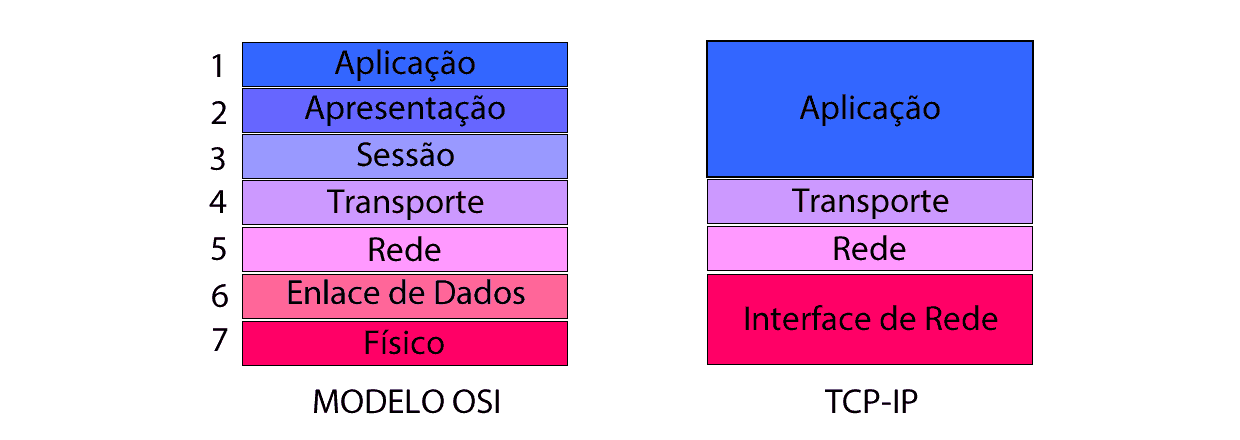
3.1.1.7. A CAMADA DE APLICAÇÃO

A Camada de Aplicação é o nível mais alto do modelo OSI, é a camada mais notável para o usuário, pois é possível interagir diretamente com ela em nosso dia-a-dia através de aplicações como os softwares.

3.1.2. TCP/IP

Em linhas gerais o TCP/IP (junção dos protocolos TCP (Transmission Control Protocol) e o IP (Internet Protocol)) é uma compilação de protocolos voltados a comunicação, foi adequado ao modelo OSI (tratado anteriormente) com o único objetivo de padronizar a comunicação de redes no mundo inteiro.

O modelo TCP/IP inicial é estruturado em 4 camadas, as quais são encarregadas de envolver as 7 camadas OSI

Arquitetura TCP/I

3.1.2.1. CAMADA DE APLICAÇÃO

A camada de aplicação exerce o papel de fazer a comunicação entre os programas e os protocolos de transporte no TCP/IP. Essa camada é conhecida pela diversidade de protocolos que trabalham nela, alguns deles são:

FTP (File Transfer Protocol): Este protocolo permitirá a transferência de arquivos entre dois computadores utilizando credenciais (login e senha).

TFTP (Trivial File Transfer Protocol): Este protocolo permitirá a transferência de arquivos entre dois computadores sem as credenciais.

SMTP (Simple Mail Transfer Protocol): Este protocolo é uma aplicação utilizada para o transporte de e-mail.

HTTP (Hypertext Transfer Protocol): Protocolo que admite transportar páginas HTML de servidores web para navegadores.

BOOTP: Protocolo que é usado para estabelecer um endereço IP a computadores que não possuam disco rígido.

O nível de Aplicação se comunica diretamente com o de transporte usando portas numeradas quais seguem um padrão para diferentes aplicações. Exemplificando, o HTTP sempre utilizará a porta 80 enquanto o FTP usará a porta 20, para transporte de dados, e 21, para controle. Esse processo permitirá ao protocolo de Transporte identificar qual a natureza do conteúdo do pacote fazendo com que o destinatário identifique qual aplicação receberá esse determinado dado.

3.1.2.2. CAMADA DE TRANSPORTE

Esta camada é responsável por capturar os dados da camada de Aplicação e dividi-los em diversos pacotes menores distintos entre si, quando há a transporte de dados no TCP/IP. Os protocolos que atuam nessa camada são:

TCP (Transmission Control Protocol): É o protocolo de conexão garantida usada inspecionar o gerenciamento das aplicações a nível de compartilhamento entre computadores. Este protocolo realiza desde o transporte em sequência até a checagem da veracidade dos dados.

UDP (User Datagram Protocol): É o protocolo de conexão não confiável usada para fiscalizar o gerenciamento das aplicações a nível de compartilhamento entre computadores. Este protocolo é utilizado na transmissão de dados quais a aplicação faz a revisão da integridade dos dados compartilhados;

ICMP (Internet control message protocol): É o protocolo que fornecerá a gerenciamento e a exposição de erros para ajudar na administração de informações no decorrer do compartilhamento entre computadores. Este protocolo é usado para reportar o status do host que está sendo conectada ao outro computador

IGMP (Internet Group Management Protocol): Usado no suporte de mensagens multicasting e rastrear grupos de usuários na rede.

3.1.2.3. CAMADA DE REDE

A camada de Rede é a encarregada por anexar o cabeçalho no pacote de dado vindo da camada de Transporte, entre os dados de controle, será adicionado o endereço IP do computador remetente e o endereço IP do computador destinatário. Entre os Protocolos que atuam nessa camada, é interessante citar:

Existem vários protocolos que trabalham na camada da Internet e podemos citar os seguintes:

ARP (Address Resolution Protocol): protocolo de mensagem que é usado para ir até a ethernet por meio de um endereço de IP, por meio dele é possível o empacotamento do dado do IP em blocos ethernet e. Sem este protocolo, o pacote de ethernet pode não ser criado do pacote de IP pois o endereço ethernet não seria deliberado.

IP (Internet Protocol): Todos os pacotes de dados dos protocolos serão encapsulados em um pacote de dados IP (exceção para ARP e RARP), o IP irá fornecer o recurso para utilizar o software para endereçar e administrar pacotes de dados sendo transmitidos pelos computadores.

RARP (Reverse address resolution): Este protocolo é utilizado com o objetivo de permitir que um computador sem uma memória não volátil tenha um endereço IP partindo do seu endereço ethernet.

3.1.2.4. CAMADA DE INTERFACE DE REDE

Para o TCP/IP todos os datagramas criados no grau de Rede serão enviados para a camada anterior, a camada de Interface de Rede, senão esta camada irá receber dados da rede e enviará para a camada de Rede. Entre os protocolos que atuam nessa camada podemos citar:

SLIP (Serial Line Internet Protocol) protocolo que posiciona os pacotes de dados em quadros de preparação para o envio pelo do hardware, sendo utilizado para enviar informações por linhas seriais. O protocolo não corrige erros, endereça ou identifica pacotes, também não autentica sendo que o mesmo suporta apenas a transmissão de pacotes IP.

CSLIP (Compressed SLIP): Este protocolo nada mais é que a compressão dos dados essenciais para o SLIP utilizando o método Van Jacobson qual é capaz de reduzir consideravelmente o volume do pacote.

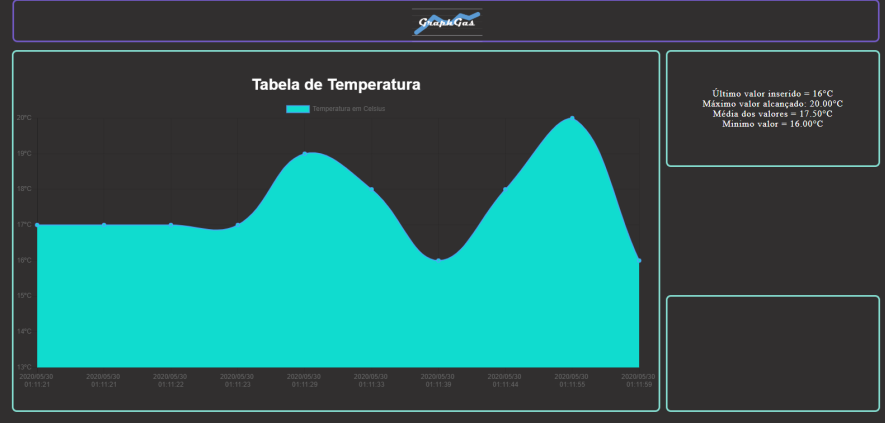
PPP (Point to Point Protocol): Este protocolo é uma maneira de transmissão de pacotes por ligações seriais, suportando linhas síncronas e assíncronas, capaz de promover uma comunicação bidirecional. Tem sido usado em substituição ao SLIP.

**4. PLANO DE DESENVOLVIMENTO DO PROJETO**

Percebendo a importância da tecnologia para a prevenção de desastres, desenvolvemos um projeto IoT totalmente voltado para ambientes industriais químicos. O projeto em si consiste em uma placa Arduino com módulo de internet embutida conectada a um sensor de temperatura, caso os níveis de temperatura se elevem de forma a ultrapassar um padrão seguro determinado pelo gestor, o mesmo conseguirá acompanhar isso diretamente pelo site. O Arduino tratará de se comunicar com um banco de dados que estará conectado a um site, o qual irá expor todo o conjunto de dados, dispostos em informações, captados pelo dispositivo.

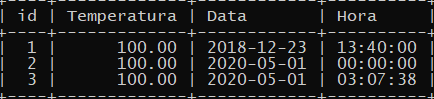
4.1.O SITE

O site de nome “Arduíno” será fundamental para que as informações capturadas pelo sensor sejam expostas para o controle de temperatura do ambiente, além disso ele apresentará as estimativas e médias diárias em um espaço reservado para isso.



Como apresentado na imagem anterior os dados formatados em informações serão apresentados em um gráfico o qual será atualizado a cada 5 segundos de acordo com as informações da placa. Ao lado pode-se perceber também uma div no canto superior direito a qual é separada para receber estimativas, como a média diária de temperatura no ambiente o ultimo valor inserido e etc.

4.2.O BANCO DE DADOS

 Para a construção do banco de dados uma única tabela no MySQL seria suficiente para assegurar que todas as informações necessárias fossem devidamente guardadas afim de que futuramente tornassem informações. Esta tabela foi denominada “temperatura” e foi estruturada da seguinte maneira:

No decorrer da construção do projeto percebemos que precisaríamos de um servidor que alocasse o banco de dados de uma maneira que ele ficasse online, para isso utilizamos o AWS.

4.2.1.AWS (AMAZON WEB SERVICES)

A Amazon é uma empresa de serviços gerais, fundada por Jeff Bezos no ano de 1994 em Seattle com o objetivo que vender livros, entretanto depois de um tempo sua área de atuação foi amplamente expandida, atualmente a empresa vende desde brinquedos até serviços de streaming.

Um dos serviços oferecidos é o intitulado AWS (Amazon Web Services), existente desde 2006 é totalmente focado em serviços de infraestrutura de TI, isso é, a chamada computação em nuvem. Esse tipo de computação dá a liberdade para deixar qualquer aplicativo, independentemente de qual seja sua natureza, em execução em nuvem. A partir desse recurso é possível hospedar aplicações quais englobem desde aplicativos até páginas web.

4.2.1.1.AWS EC2

O Amazon EC2 (Elastic Compute Cloud) é uma subpartição do AWS que tem o objetivo de facilitar a obtenção dos chamados servidores virtuais, ou seja, é permitida a criação de máquinas virtuais de forma facilitada, muito segura e com controle garantido sobre todos os seus recursos.

A partir dessa ferramenta e habilitando o acesso de externo conseguimos permitir que, não somente, o nosso banco de dados ficasse totalmente online como também o site, assim o database estaria apto para receber os dados diretamente do dispositivo Arduino.

4.3.A API

Com a necessidade de fazer com que a página web capturasse as informações da tabela “temperatura” no banco de dados, vimos a necessidade de criar uma API. Este recurso será totalmente responsável por conectar ao banco de dados ao site partindo de uma requisição feita pelo front-end, ela também será encarregada de retornar dados formatados como objetos.

Para a criação dessa API utilizamos o Framework Express, ele é capaz de criar a estrutura com rotas bem definidas possibilitando a utilização de requisições HTTP tendo como exemplo a requisição de método get que será utilizada ao longo do projeto. O Express utiliza TypeScript como linguagem. Essa linguagem é baseada em JavaScript com a adição de recursos importantes, como possibilitar o uso de programação orientada a objetos. O Framework é instalado juntamente aos seus módulos, um desses módulos é o mysql. A pasta em nosso projeto dedicado a API é dividida em alguns arquivos, os mais importantes são:

O Arquivo mysql.js: Este arquivo algumas contém variáveis, dentre estas se localiza a variável “pool” que é composta pelo método create pool da variável mysql nesse método são colocadas algumas informações relativas ao banco de dados como o usuário, o nome e o host alocado, após isso utilizamos um método do framework chamado exports.pool recebendo a esta variável.

Pasta Routes: dentro dessa pasta poderemos encontrar o arquivo temperatura.js, dentro desse arquivo se localiza a variável router. Esta variável será definida pelo método .router() qual receberá como parâmetro a função express.Router() esse método será responsável por criar os métodos de requisição HTTP.

Após isso é criado o método http get dentro da rota temperatura através da anteriormente declarada variável router dispondo do seu método .get. Os parâmetros disponibilizados para este método consistem em: especificação de onde está o arquivo utilizável e como segundo parâmetro recebe uma função anônima. Esta função exige 3 parâmetros o primeiro é definido pelo tipo de requisição o segundo será a natureza do arquivo que será retornado e enfim a especificação do que será realizado em seguida.

No decorrer dessa função será invocada a variável mysql para utilização do seu método .getConnection o parâmetro deste método é definido por outra função anônima com dois fatores o inicial é o de erro que só será utilizado caso haja alguma falha na comunicação com o banco de dados uma estrutura if testará se o erro for verdadeiro a API deverá enviar um status HTTP de número 500 que significa que ocorreu um erro no servidor. O segundo parâmetro é definido pela variável “conn” que caso a conexão seja realizada com sucesso guardará os comandos SQL que serão efetuados no bancos de dados, após esta condição utilizando parâmetro “conn” é realizada uma chamada através do método query qual receberá como primeiro parâmetro a sintaxe em sql que deverá ser efetuada no database e por fim é recebida outra função anônima com três parâmetros, sendo eles um de erro qual será retornado do banco de dados em caso falha utilizando uma condição similar com anterior que verifica se o parâmetro é verdadeiro e caso seja retornará um status HTTP de número 500 que significa falha no comando sql o segundo receberá o retorno do banco de dados em caso de sucesso e o ultimo nomeado “fields” será utilizado para fazer uma busca precisa nas colunas da tabela.

Por seguinte um laço de repetição “for” que girará em torno do resultado recebido pelo banco de dados e inserirá os valores de cada linha capturada no banco de dados em variáveis do tipo vetor. Após esse processo é retornado o status de número 200. Isso significa que o procedimento ocorreu sem nenhuma falha. Logo em seguida passará como retorno um objeto contendo todas as variáveis vetor definidas anteriormente.

O último recurso desse arquivo solicitará o encerramento da conexão com o database por meio do método release da variável “conn”, além disso é solicitado através do método do framework module.exports a exportação de todo o conteúdo da variável Router.

Além do descrito a API conta com o arquivo app.js que tratará de algumas outras configurações importantes.

4.4.ARQUIVOS JAVASCRIPT

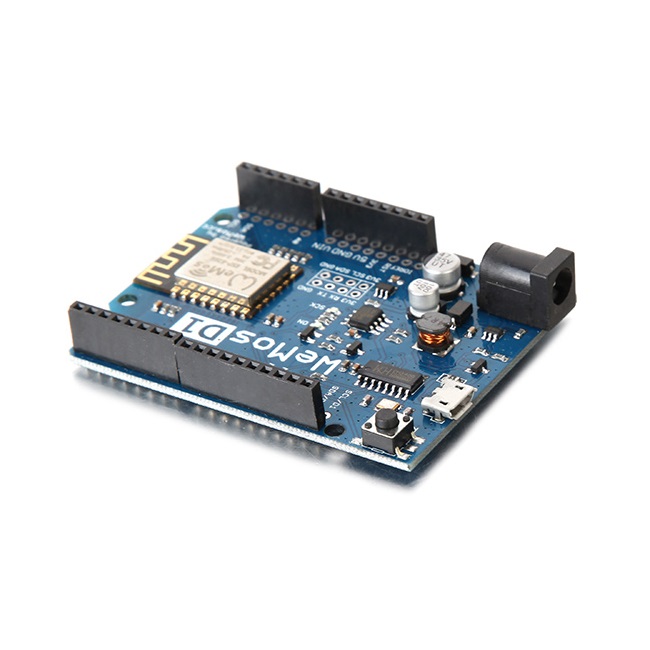
Além de todos os elementos descritos o projeto utiliza de mais dois recursos Java Script, os arquivos Chart.js, responsável pela conexão do front-end com o back-end, além disso ele desenvolve um papel importante na geração e atualização do gráfico e por fim o arquivo Index.js qual será responsável por mudar atualizar os dados na div superior direita.

O repositório github do projeto se encontra no link https://github.com/MarllonCampos/APS-5.

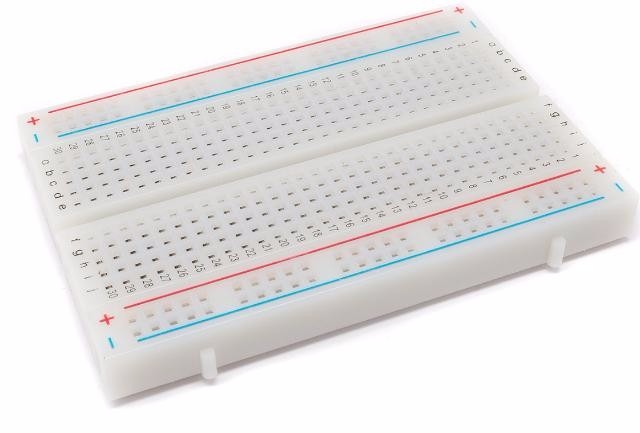
**5. O** **PROJETO**

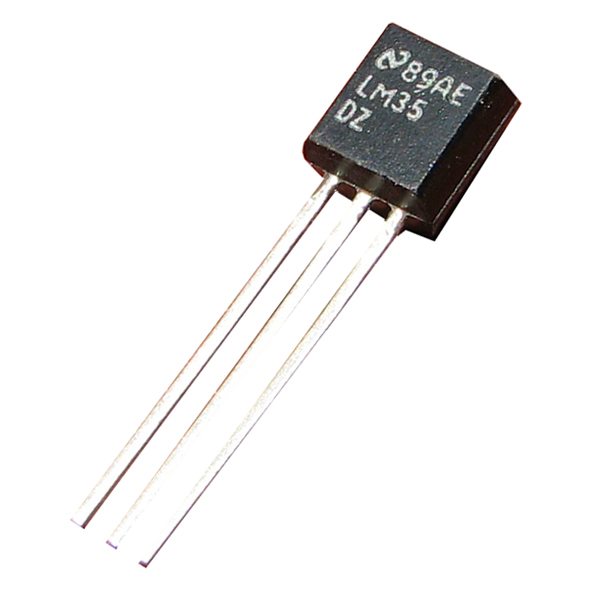
Para que o dispositivo fosse finalmente concretizado foram utilizados alguns recursos físicos, estes são:

5.1 COMPONENTES

 Placa WeMos D1 R1 WiFi – Esta placa tem processador ESP8266-12E que pode operar em 80MHz / 160MHz conectividade WiFi nativa padrão 802.11b/g/n, sendo excepcionalmente indicada em projetos de natureza IoT. A estrutura possui conector micro USB, 11 pinos de I/O digitais e 1 entrada analógica de no máximo 3.3V, além disso sua programação é compatível com Arduino e NodeMCU. Possui arquitetura RISC 32 bits com 4Mb de memória flash.

Protoboard 400 pontos – Ferramenta extremamente útil para testes utilizando componentes eletrônicos estrutura consiste em uma placa contendo uma matriz de contatos que dá liberdade para a construção de circuitos experimentais sem a precisar de solda, ou seja, permite com rapidez e segurança alterar ou substituir um determinado componente de posição de um componente até sua substituição. Através dela podemos conectar componentes como capacitores, resistores e transistores.



Sensor de temperatura LM35DZ - um sensor qual apresenta uma saída de tensão linear de acordo à temperatura na qual o próprio dispositivo se encontrar em determinado instante, tem em sua saída um sinal de 10mV para cada Grau Célsius dentro da faixa de temperatura entre –55ºC e 150ºC.

Jumper Macho-Macho – Através dos jumpers é possível realizar conexões entre componentes eletrônicos, suas extremidades são isoladas tornando as conexões anatomicamente seguras.

5.2. MONTAGEM

Para que o dispositivo funcionasse como devido foi necessário, obviamente, que fizéssemos as ligações dos componentes de forma concisa. O primeiro passo foi ligar a protoboarb no pino de 5 volts da placa Arduino e ao terra da mesma, após isso conectamos sensor LM35 qual possui três pinos, o primeiro é o de alimentação com 5 volts, o segundo ao terra do Arduino e a última na porta analógica esta será responsável por ler os bits devolvidos pelo sensor.

Como o LM35 é um transistor assim que entender a mudança de mV ele subirá um °C, precisaremos ajustar os valores obtidos usando a formula que captura o valor lido pelo sensor divide por 1024 e em seguida multiplica por 5000mV dividindo todo o valor por 10.

5.3. CONECTANDO COM O BANCO DE DADOS

É imprescindível que o dispositivo guarde as informações coletadas sejam armazenadas no banco de dados, para isso precisamos conecta-los usando programação em Arduino que é baseada em c++.

O Arduino tem dois métodos padrão, o setup, que é utilizado para depositar comandos que serão executados logo ao ligar a placa, por exemplo informar onde se encontra um led, sensor ou conexões externas e o método loop que é responsável por abrigar comandos que serão executados a todo momento em que o dispositivo estiver ligado, por exemplo definir que um led piscará a cada 5 segundos. Para o nosso caso precisamos importar bibliotecas externas como a de WiFi “ESP8266Wifi.h”.

O código no geral consiste na tentativa de conexão com WiFi local, caso não obtenha êxito entrará num loop até que estabeleça uma conexão sólida, logo em seguida o método do mysql fará a mesma coisa até conseguir comunicação com o banco de dados. Conectado a internet e comunicando-se com o banco de dados restará apenas entrar em um loop o qual lerá a informação captada pelo sensor e enviará direto ao banco de dados.

**6. RELATÓRIO DESCRITIVO DO PROJETO**

Recapitulando, o projeto apresentado consiste em criar uma solução IOT eficaz para ambientes industriais químicos, com a intenção de prevenir grandes incêndios. Para isso criamos um dispositivo com a placa Arduino que monitorará a temperatura do ambiente usando um sensor de temperatura LM35. As temperaturas capturadas serão enviadas para um banco de dados online que armazenará todos os dados relevantes. Para que as informações sejam dispostas de forma dinâmica, criamos um site que se comunicará com o banco de dados através de uma API, com esse recurso o gráfico alocado na página web poderá ser alimentado mostrando a variação de temperatura nos determinados dias e seus respectivos horários.

6.1. IMPACTO AMBIENTAL E SOCIAL DO PROJETO

O risco ambiental relacionado a empresas químicas é intrínseco pois a natureza desse ramo produtivo inclui elementos tóxicos a qualquer ser vivo dês de matérias primas a tipos de resíduos gerado. Podemos citar inúmeros acidentes ambientais que envolvem indústrias químicas no decorrer da história desse ramo, e em diversos países. Esses acidentes podem acontecer por diversos motivos como falhas relacionadas ao processo produtivo, forma de armazenamento, transporte ou manipulação de componentes.

Pelos motivos apresentados é indubitável que esse tipo de fabrica apresenta um elevado risco a seres humanos, não somente de pessoas internas a indústria em questão, mas também a população das redondezas, uma vez que caso haja um incêndio os gases e substancias toxicas irão se dissipar de diversas formas, seja pela própria fumaça ou até mesmo por uma eventual chuva causada por nuvens infectadas.

É imprescindível que cuidados sejam tomados em cada detalhe minucioso nesse ramo, por esse motivo o nosso projeto teria um impacto positivo para a prevenção de acidentes dessa vertente, diminuindo a probabilidade não somente de perdas humanas como também patrimônios físicos.

6.2. POLITICA DE SEGURANÇA

Como o nosso projeto é voltado diretamente para a gerencia, manteremos a confidencialidade dos dados, garantindo que os acessos às informações sejam permitidos apenas às pessoas autorizadas, isso é gestores e colaboradores ligados a segurança da empresa.

Em relação a externos o fato de toda a estrutura lógica estar alocada em nuvem dá a segurança de que existem pouquíssimas chances de haver um ataque externo, Toda configuração poderá ser feita pelo próprio gerenciamento de ferramentas de permissão do próprio AWS além da permissão de entrada de dados no próprio banco de dados é configurado no, então citador, servidor localizado na nuvem.

6.3. ANÁLISE DE CUSTOS

Quando falamos em prevenção de vidas humanas é importante que não haja margem para erros por isso é imprescindível que um grande investimento financeiro seja aplicado nesta área. Fazendo a análise de custos para a implementação no nosso projeto, considerando a faixa de preço de cada elementos:

- Placa WeMos D1 R1 WiFi: Preço entre R$45,00 e 70,00;

- Protoboard 400 pontos: Preço entre R$10,00 e 15,00;

- Sensor de temperatura LM35DZ: Preço entre R$9,00 e 15,00;

- Jumper Macho-Macho: Preço entre R$9,00 e 13,00;

Desconsiderando gastos com recursos lógicos o orçamento do hardware está entre R$73,00 e R$113.

**7. CONSIDERAÇÕES FINAIS**

Como destacamos durante todo o desenvolvido deste trabalho, é notável a importância da tecnologia para todas as áreas humanas, percebemos que ela tende a crescer progressivamente em prol da prevenção de desastres, acarretando em si uma maior segurança, para desde colaboradores das empresas até o meio ambiente.

Considerando que, por mais ampliado que seja, o conhecimento humano está passível a erros, investir na tecnologia para a segurança humana é a coisa mais inteligente a ser feita. Graças a utilização de dispositivos como o que desenvolvemos, é permitido detectar os lugares e os afazeres com maior índice de acidentes, tornando mais rápido a tomada de medidas imprescindíveis aos diversos casos.

Como consequência construtiva, o ambiente torna-se confiável para as pessoas envolvidas, trazendo maior produtividade para a empresa.

**REFERENCIAS**

DAMASCENO, Paulo Augusto**. CAUSAS E RISCOS DE INCENDIO EM AMBIENTES INDUSTRIAIS**. Disponível em: <http://blog.mifire.com.br/causas-e-riscos-de-incendio-em-ambientes-industriais/ > Acesso em: 10 de Maio de 2020.

NEIVA, Sílvia Beatriz**. INTRODUÇÃO A COMUNICAÇÃO DE DADOS**. Disponível em: < https://www.gta.ufrj.br/grad/silvia/silvia.htm > Acesso em: 11 de Maio de 2020.

GAIDARGI, Juliana. **O QUE É TCP/IP E COMO FUNCIONA.** Disponível em: <https://www.infonova.com.br/artigo/o-que-e-tcp-ip-e-como-funciona/ >Acesso em: 15 de maio de 2020.

GOMES, Pedro César Tebaldi. **CONHEÇA OS 20 PRINCIPAIS SERVIÇOS DA AWS: AMAZON WEB SERVICES.** Disponível em: < https://www.escolalinux.com.br/blog/o-que-e-aws-pra-que-serve-e-por-que-devo-domina-la>Acesso em: 17 de maio de 2020.

OLIVEIRA, Paulo. **O QUE É AWS, PRA QUÊ SERVE E POR QUE DEVO DOMINA-LA.** Disponível em: < https://www.escolalinux.com.br/blog/o-que-e-aws-pra-que-serve-e-por-que-devo-domina-la>Acesso em: 17 de maio de 2020.

SANTOS, André**. REDES DE COMUNICAÇÃO DE DADOS**. Disponível em: < https://www.uniaogeek.com.br/redes-de-comunicacao-de-dados-principais-conceitos/> Acesso em: 11 de Maio de 2020.

SILVEIRA, Cristiano Bertulucci. **ENTENDA COMO FUNCIONA O PROTOCOLO TCP/IP.** Disponível em: < https://www.citisystems.com.br/protocolo-tcp-ip/ >Acesso em: 15 de maio de 2020.

**O MODELO TCP/IP.** Disponível em: < https://paginas.fe.up.pt/~mrs01003/TCP\_IP.htm > Acesso em: 15 de maio de 2020.

**O QUE É EC2 AWS.** Disponível em: < https://skyone.solutions/pb/o-que-e-ec2-aws/> Acesso em: 17 de maio de 2020.

**TECNOLOGIA A SERVIÇO DA PREVENÇÃO E RECUPERAÇÃO DE DESASTRES NATURAIS**. Disponível em: < https://www.revistaplaneta.com.br/tecnologia-a-servico-da-prevencao-e-recuperacao-de-desastres-naturais/> Acesso em: 05 de maio de 2020.

**CÓDIGO FONTE**

Index.html:

 <!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="Content-Security-Policy"

        content="default-src \*;

               script-src 'self' 'unsafe-inline' 'unsafe-eval'

                           127.0.0.1:\*

                           http://\*.gstatic.com

                           http://\*.googleapis.com

                           https://\*.gstatic.com

                           https://\*.googleapis.com

                           https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.min.js

                           ;

               style-src  'self' 'unsafe-inline'

                           127.0.0.1

                           http://\*.gstatic.com

                           http://\*.googleapis.com

                           https://\*.gstatic.com

                           https://\*.googleapis.com

    ">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Arduino</title>

    <link class='pagestyle 'rel="stylesheet" href="css/style.css">

</head>

<body>

   <div class="container">

       <div class="logo">

          <div id="teste" >

          </div>

       </div>

        <div class="container-content">

            <div class='content'id='center'>

                <div class="grafico">

                    <canvas class="chart"></canvas>

                </div>

            </div>

            <div class='content' id='upper-right'>

                Último valor inserido = Ultimo Valor

                Máximo valor alcançado = Máxima <br>

                Média dos valores = Médias <br>

                Minimo valor = Minímo<br>

            </div>

            <div class='content' id='lower-right'></div>

        </div>

    </div>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.min.js"></script>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

    <script src="js/chart.js"></script>

    <script src="js/index.js"></script>

</body>

</html>

**Chart.js**

var ctx = document.getElementsByClassName('chart')

var temperatura =[]

var data\_hora  = []

var grafico = new Chart(ctx,{

    type:'line',

    data:{

        labels:['a30',"b40","c60","d70"],

        datasets:

            [{label:"Temperatura em Celsius",

               data:["30","33","21","12"],

            borderWidth:2,

            borderColor:'rgba(77,166,253,0.85)',

            backgroundColor:'#10dccf',

        }]

    },

    options:{

        title:{

            display:true,

            text: "Tabela de Temperatura",

            fontSize:28,

            fontColor:'white',

            fontWeight:'bold',

        },

        scales:{

            yAxes:[{

                ticks:{

                    callback: function(value,index,values){ return  value + 'ºC'},

                    suggestedMin:13

            }}],

        }

    }

})

var RefreshAutomatico = setInterval(function minhafuncao() {

    $.ajax({

         url: 'http://18.228.194.165:3000/temperatura/',

         dataType: 'json',

         cache:false,

         type: 'get',

         success: function (response) {

             updateChart(response)

         },

         error:function(erro){

             console.log("Alerta, erro em conectar a API")

         },

     })

    function updateChart(data){

        grafico.data.labels = []

        grafico.data.datasets[0].data = []

        var k = 10

        if (data.temperatura.length > 10 ){

            for (let i = 0 ; i < data.temperatura.length - (data.temperatura.length -10) ; i++){

                let j = data.temperatura.length - k

                let tempoEdata = (`${(data.data[j].substring(0,10).replace("-",String.fromCharCode(47))).replace("-",String.    fromCharCode(47))} \*${data.hora[j]}`).split('\*')

                grafico.data.labels.push(tempoEdata)

                grafico.data.datasets[0].data.push(data.temperatura[j])

                k = k - 1

            }

        }else{

            for (let i = 0 ; i < data.temperatura.length ; i++){

                let tempoEdata = (`${(data.data[i].substring(0,10).replace("-",String.fromCharCode(47))).replace("-",String.        fromCharCode(47))} \*${data.hora[i]}`).split('\*')

                grafico.data.labels.push(tempoEdata)

                grafico.data.datasets[0].data.push(data.temperatura[i])

            }

        }

        grafico.update()

        window.updateMedia() // Chama essa função do arquivo "index.js"

    }

}   ,3000)

**Index.js**

var upper\_right = document.getElementById("upper-right")

var max = 0

var media =0

var min = 0

var lastValue

function updateMedia (){

    const temperatura = window.grafico.data.datasets[0].data.slice()

    max = temperatura[0]

    min = temperatura[0]

    media = 0

    for(let i = 0; i < temperatura.length; i++){

        if(temperatura[i] > max){

            max = temperatura [i]

        }

        if(temperatura[i] < min){

            min = temperatura[i]

        }

        media += temperatura[i]

        lastValue = temperatura[temperatura.length - 1]

    }

    media = media/temperatura.length

    upper\_right.innerHTML = `Último valor inserido = ${lastValue}°C <br>Máximo valor alcançado: ${max.toFixed(2)}°C <br> Média dos valores = ${media.toFixed(2)}°C <br> Minimo valor = ${min.toFixed(2)}°C `

}

**Style.css**

body{

    background-color: #312f2f;

    padding: 0px;

    margin:0;

}

.container{

    margin: auto;

    margin-top: 1vh;

    display: grid;

    grid-template-rows: 2;

    grid-gap:12px ;

}

/\* Criação do GRID e estilo dos objetos dentro do grid\*/

.container-content{

    width: 97vw;

    height: 82vh;

    margin:auto;

    margin-top: 0.3vh;

    display: grid;

    grid-template-columns: 1fr 2fr 1fr ;

    grid-template-rows: 1fr 1fr 1fr;

    text-align: center;

    grid-gap:10px;

    color:black;

}

.content {

    border: 3px solid #84dccf;

    border-radius: 9px;

}

/\*Fim da criação do GRID e estilos\*/

.logo{

    border: 3px solid #7159c1;

    border-radius:9px;

    width: 96.5vw;

    height: 9vh;

    margin: auto;

    display: grid;

    grid-template-columns:1fr 1fr 1fr;

    grid-template-rows: 1fr;

    align-items: center;

    justify-content: center;

    text-align: center;

}

canvas{

    width: 100%;

    height: 100%;

}

#teste{

    grid-column: 2/3;

    grid-row: 1/1;

    content: url("GraphGas.png");

    max-width:30%;

    max-height: 90.99%;

    justify-self: center;

}

/\* Ajustes no GRID \*/

#upper-right{

    grid-column: 3/4;

    grid-row:1/2;

    display: flex;

    align-items: center;

    justify-content: center;

    text-justify: auto;

    letter-spacing: 1px;

    color:white;

}

#lower-right{

    grid-column:3/4;

    grid-row: 3/4;

}

#center{

    grid-column: 1/3;

    grid-row: 1/4;

    display:grid;

    align-items: center;

}

/\*Fim dos Ajustes no GRID\*/

**Arduinocode.ino**

#include <WiFiClient.h>

#include <ESP8266WiFi.h>

#include <ESP8266WebServer.h>

#include <ESP8266mDNS.h>

#include <MySQL\_Connection.h>

#include <MySQL\_Cursor.h>

#include <MySQL\_Encrypt\_Sha1.h>

#include <MySQL\_Packet.h>

//declaracao wifi

char\* rede="AMOR E PAZ"; //Nome da WIFI

char\* senha="xeparacomisso"; //SEnha da wifi

ESP8266WebServer server(80);

//declaracao de conexão do ip BD

IPAddress server\_addr(85, 10, 205, 173);

char senhab[]="lavs1998"; //senha usuario bd

char user[]="mingosso1"; //nome do usuarido do bd

char insert\_sql[]="INSERT INTO TEMPERATURA (Temperatura,Data,Hora) VALUES ('%d','curdate()','curtime()')"; // Query MYSQL

char banco[]="temperatura1";

char query[128];

//instalando objetos

ESP8266WebServer client;

MySQL\_Connection con((Client \*)&client);

// declaracao de variaveis

int temperatura;

unsigned long lm35;

int tempo =0 ;

int tempoAntigo =0;

int espera = 3000;

//incio do setup

void setup()

{

  Serial.begin(9600);

  pinMode(A0,INPUT);

  wifi();

  mysql();

}

void loop()

{

 temperatura=analogRead(A0);

 lm35=(20+0.488155\*(temperatura-20));

 if(tempo - tempoAntigo > espera){

    Serial.println("Enviando o valor da tempertaura ");

    enviar();

    tempoAntigo = tempo;

}

delay(10);

}

//Enviar para o BD

void enviar(){

  sprintf(query,insert\_sql,lm35);

  MySQL\_Cursor \*cur\_nem = new MySQL\_Cursor(&con);

  cur\_nem->execute(query);

  delete cur\_nem;

  Serial.print("informacoes enviadas");

}

//conexao com a wifi

void wifi(){

  digitalWrite(LED\_BUILTIN,LOW);

  WiFi.begin(rede,senha);

  while(WiFi.status()!=WL\_CONNECTED){

    delay(500);

  }

  server.begin();

  Serial.println("conectado a wifi");

  Serial.print("IP: ");

  Serial.print(WiFi.localIP());

  Serial.println();

  digitalWrite(LED\_BUILTIN,HIGH);

}

//conecta mysql

void mysql(){

  digitalWrite(LED\_BUILTIN,LOW);

  while(!Serial){

    ;

  }

  while(!con.connect(server\_addr,3306,user,senhab)){

    Serial.println("conexao falhou");

    con.close();

    delay(1000);

    Serial.println("tentando novamente");

  }

  Serial.println("conectado ao servidor sql");

  digitalWrite(LED\_BUILTIN,HIGH);

}

**Mysql.js**

const mysql = require('mysql')

const pool = mysql.createPool({

            "user":"marllon",

            "password":"Marllon.10",

            "database":"aps",

            "host":"localhost",

            "port":"3306"

})

exports.pool = pool

**app.js**

const express = require('express')

const app = express()

const morgan = require('morgan')

const bodyParser = require('body-parser')

const router = express.Router()

const mysql = require('mysql').pool

const Cors = require('cors')

const rotaTemperatura = require('./routes/temperatura')

app.use(morgan('dev'))

app.use(bodyParser.urlencoded({extended:false}))

app.use(bodyParser.json())

app.use(Cors())

app.use((req,res,next)=>{

    if(req.method === 'OPTIONS'){

        res.header('Acess-Control-Allow-Methods','PUT,POST,PATCH,DELETE,GET');

        return res.status(200).send({});

    }

    next();

})

app.get('/',(request,response)=>{

    response.status(200).send({

        mensagem : "Hello World"

    })

})

app.use('/temperatura',rotaTemperatura)

module.exports = app

**temperatura.js**

const express = require('express')

const router = express.Router()

const mysql = require('../mysql').pool

var temperatura = []

var data = []

var hora = []

var inserir = []

router.get('/',(req,res,next)=>{

    mysql.getConnection((error,conn)=>{

        if(error){return res.status(500).send({error:error})}

        conn.query(

            "select \* from TEMPERATURA",

            (error,resultado,fields)=>{

                if(error){return res.status(500).send({error:error})};

               for(let i = 0 ; i < resultado.length; i++){

                   temperatura.push(resultado[i].Temperatura)

                   data.push(resultado[i].Data)

                   inserir[i] = (Number((resultado[i].Hora).substr(0,2)) - 3 )

                   if ( inserir[i] < 10){

                       inserir.push((inserir[i]))

                       inserir[i] = "0"+inserir[i]

                   }

                   inserir[i] = inserir[i].toString()

                   hora.push(`${inserir[i]}${(resultado[i].Hora.substr(2,7))}`)

               }

                return res.status(200).send(

                    {

                        temperatura: temperatura,

                        hora:hora,

                        data: data

                    }

                );

            })

            temperatura = []

            hora = []

            data = []

        conn.release();

    })

    console.log(req.ip)

})

module.exports=router

**FICHAS DE ATIVIDADE PRÁTICA SUPERVISIONADA**

