

UNIVERSIDADE PAULISTA
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGIA – ICET
CURSO DE CIÊNCIA DA COMPUTAÇÃO

**DESENVOLVIMENTO DE SOLUÇÃO UTILIZANDO REDE DE
COMUNICAÇÃO DE DADOS E DISPOSITIVOS IOT (INTERNET OF
THINGS)**

SÃO PAULO
2022

BRUNO GONZALEZ MASSONE – N582JF3
FELIPE DE OLIVEIRA PEREIRA MAURÍCIO – F2322A8
FELIPE NUNES DA SILVA – N4935E0

DESENVOLVIMENTO DE SOLUÇÃO UTILIZANDO REDE DE COMUNICAÇÃO DE DADOS E DISPOSITIVOS IOT (INTERNET OF THINGS)

Atividades Práticas Supervisionadas – APS -
DESENVOLVIMENTO DE SOLUÇÃO
UTILIZANDO REDE DE COMUNICAÇÃO DE
DADOS E DISPOSITIVOS IOT (INTERNET OF
THINGS) – trabalho apresentado como
exigência para a avaliação do 5º semestre, do
curso de **Ciência da Computação** da
Universidade Paulista sob orientação de
professores do semestre.

Orientador: João Fernandes
(Arquitetura de Redes de Computadores)

SÃO PAULO
2022

LISTA DE FIGURAS

Figura 1- Modelo OSI	13
Figura 2 - Modelo OSI e TCP/IP.....	15
Figura 3 - ESP32.....	18
Figura 4 - Arquitetura ESP32	19
Figura 5 - Protoboard	20
Figura 6 - Cabos de conexão da protoboard.....	20
Figura 7 - Funcionamento do sensor ultrassônico	21
Figura 8 - Esquema elétrico do dispositivo.....	21
Figura 9 - Dispositivo montado.....	22
Figura 10 - Instalação de bibliotecas adicionais	23
Figura 11 - Funções Principais.....	24
Figura 12 – Funções Multithread.....	24
Figura 13 - Manipulação de objetos JSON.....	25
Figura 14 - Cálculo de distância	25
Figura 15 - Rotas.....	26
Figura 16 - Função de conexão.....	26
Figura 17 - Função de criação do objeto JSON	27
Figura 18 - Funcionamento do dispositivo.....	27
Figura 19 - Python e Flask	28
Figura 20 - Arquivo main.py	29
Figura 21 - Chart.js.....	29
Figura 22 - Visualização dos dados (condições normais)	30
Figura 23 - Visualização de dados (alerta de enchente)	31
Figura 24 - Alterar parâmetros	32
Figura 25 - Dispositivos.....	32
Figura 26 - Diagrama lógico	33
Figura 27 - Topologia de rede	35
Figura 28 - Funções de Thread	38
Figura 29 - Arquivo de retorno JSON	38

SUMÁRIO

1. OBJETIVO	5
2. INTRODUÇÃO	6
3. FUNDAMENTOS DA COMUNICAÇÃO DE DADOS EM REDE.....	11
3.1 História da Internet.....	11
3.2 Redes de Computadores	12
3.2 Modelos, Protocolos e Serviços de rede.....	13
3.2.1 Modelo OSI (<i>Open Systems Interconnection</i>)	13
3.2.2 Modelo TCP/IP	14
3.2.3 Protocolos de Rede.....	16
4. DESENVOLVIMENTO	18
4.1 Parte Física (Hardware)	18
4.2 Parte Lógica.....	22
4.2.1 Desenvolvimento do Firmware do Dispositivo (<i>Software</i>)	22
4.2.2 Software de visualização dos dados	27
5. ESTRUTURA DO PROJETO.....	33
5.1 Diagrama Lógico.....	33
5.2 Topologia	34
5.3 Componentes Utilizados	35
6. RELATÓRIO	37
8. CONCLUSÃO	41
REFÊRENCIAS BIBLIOGRAFICAS	42
FICHAS DE ATIVIDADES PRÁTICAS SUPERVISIONADAS.....	43

1. OBJETIVO

A Atividade Prática Supervisionada (APS) do 5º Semestre tem como principal objetivo a criação de um sistema IOT (*Internet of Things*) para o monitoramento de desastres ambientais e a visualização dos dados captados pelos sensores através de páginas web ou de aplicativos.

As aulas ministradas durante o semestre letivo, serviram como base para o desenvolvimento de um sistema embarcado utilizando um dispositivo ESP32 e um sensor ultrasônico para captação de dados. Escrito em C, foi criado um programa capaz de medir o nível de água em solos e assim evitar danos e mortes causados por enchentes e alagamentos.

Com o que foi aprendido durante as aulas de redes de computadores foi possível interligar e integrar esse sistema com um servidor local e até mesmo com a internet, o dispositivo em questão pode ser acessado remotamente se configurado para tal finalidade.

Através de discussões sobre o tema e de pesquisas relacionadas, o grupo conseguiu compreender os danos que os desastres ambientais podem trazer à sociedade, bem como a compreensão de como sistemas IOT funcionam e como podem nos ajudar no dia a dia com tarefas simples, mas que se tornam difíceis devido ao nosso cotidiano.

Ao decorrer desse trabalho, pretendemos mostrar como o dispositivo pode nos ajudar, suas limitações e suas praticidades.

2. INTRODUÇÃO

Os Desastres Naturais representam um conjunto de fenômenos que fazem parte da geodinâmica terrestre, portanto, da natureza do planeta. Quando ocorrem, podem trazer conseqüências catastróficas para o ser humano e, por mais que a tecnologia na área seja avançada, muitos desastres naturais são imprevisíveis.

Esses fenômenos naturais representam a mudança de ciclo na Terra. No entanto, nos tempos atuais, essas ocorrências têm aumentado de maneira significativa, o que nos leva a crer nas estatísticas e estudos sobre o meio ambiente.

Nesse sentido, muitos desastres têm ocorrido porque o planeta Terra está sofrendo cada vez mais com o aquecimento global e o efeito estufa, resultando no aumento dos desastres naturais, ocasionados pelo desequilíbrio da natureza.

Para os seres humanos, muitos danos e prejuízos são resultantes dos desastres naturais, os quais geram diversos impactos na sociedade.

Por sua vez, para a natureza, os desastres naturais auxiliam na renovação e manutenção dos ecossistemas, formação do relevo, abastecimento das fontes hídricas naturais, dentre outros. Entre os desastres naturais podemos destacar:

- **Tempestades:** São tempestades de chuvas, neve, granizo, areia, raios e podem ser altamente destrutivas, dependendo da quantidade precipitada (chuvas torrenciais) e da força que apresentam. Podem levar a situações catastróficas, como: o deslizamento de terras, de gelo, caída de árvores ou torres de energia, dentre outros.
- **Terremotos (Sismos) e Maremotos (Tsunamis):** Também chamados de abalos sísmicos, representam fenômenos de vibração brusca e passageira da superfície da Terra que ocorrem por meio da movimentação das placas rochosas, bem como da atividade vulcânica e dos deslocamentos de gases no interior da Terra. Os maremotos ou tsunamis são os terremotos que acontecem dentro dos mares, provocando imensas deslocções de água.
- **Furacões, Ciclones e Tufão:** Fenômenos intensificados pelas massas de ar e dependendo da força que atingem podem arrasas cidades inteiras.
- **Seca:** Intensificada nos últimos anos com o aquecimento global, a seca tornou-se um problema enfrentado por muitos grupos pelo mundo. Dessa forma, as alterações climáticas demonstram que diversas foram as conseqüências das ações humanas durante séculos no planeta, gerando problemas como a seca e conseqüentemente a expansão do processo de desertificação.

- **Erupções Vulcânicas:** As erupções vulcânicas são perigosas na medida em que a lava expelida pelos vulcões é tão quente que pode destruir comunidades, vegetais e animais, dependendo do local que atuam.
- **Inundações:** As inundações ou enchentes são fenômenos da natureza, intensificados pela ação humana e que vem aumentando de maneira significativa nas últimas décadas. Um exemplo é o excesso de lixo, os quais entopem os bueiros, impedindo a passagem de água. As enchentes e inundações, causadas pelo aumento de quantidade das chuvas e impedimento da evacuação, provocam desabamentos que podem levar a morte de milhares de pessoas, além de grande destruição.

Alguns dos principais desastres naturais que marcaram o mundo na atualidade:

- **Sismo e Tsunami na Indonésia:** Em 26 de dezembro de 2004, um terremoto de magnitude 9 devastou grande parte da costa oeste de Sumatra, na Indonésia. O terceiro maior maremoto do mundo atingiu cerca de quinze países da região, resultando na morte de mais de 230 mil pessoas.
- **Furacão Katrina:** Em 29 de agosto de 2005, nos Estados Unidos, surge um enorme furacão de categoria 5, responsável por destruir parte da região litorânea sul do país. A velocidade dos ventos ultrapassou 280 quilômetros por hora e resultou na morte de duas mil pessoas.
- **Terremoto do Haiti:** Em 12 de janeiro de 2010, Porto Príncipe, a capital do Haiti foi atingida por um terremoto de magnitude 7, levando a morte de mais de 200 mil pessoas.

As mudanças climáticas globais atingem todo o planeta, sendo o Brasil um dos países que estão inclusos na lista, visto que ultimamente apresentam um grande aumento das ocorrências de desastres naturais por todo o país.

Além da seca que assola as regiões norte e nordeste do país, a intensificação das precipitações, junto aos fenômenos climáticos, por exemplo, o “El Nino”, demonstram o aumento das temperaturas do índice pluviométrico (chuvas) e tempestades, resultando em diversas catástrofes por todo o país.

De tal maneira, enquanto as regiões do norte e nordeste sofrem com a estiagem, as regiões sudeste e sul, no mesmo momento, sofrem com o aumento das chuvas, levando ao aumento dos alagamentos e desabamentos.

A maioria dos desastres no Brasil (cerca de 80%) está intimamente relacionada às instabilidades atmosféricas, responsáveis pelo desenvolvimento dos desastres naturais, dos quais estão as inundações, vendavais, tornados, granizos e deslizamentos de terra.

As enchentes são fenômenos prioritariamente naturais. Elas são provocadas especialmente por grandes eventos chuvosos, em termos de quantidade e/ou constância. Assim, o acúmulo de água da chuva provoca o transbordamento dos cursos de água. Esse extravasamento é um processo natural nas zonas de inundação dos rios, chamadas de vales fluviais ou planícies de inundação. As enchentes ocorrem de forma periódica, com destaque para as épocas mais chuvosas do ano.

Não obstante, a ação do homem no espaço natural potencializa esse fenômeno. Nesse contexto, a ocupação do espaço natural e a sua transformação pela ação antrópica provocam o aumento da intensidade das enchentes. Sendo assim, com base no contexto humano, as enchentes são provocadas, entre outros, pela ocupação desordenada do espaço, pela impermeabilização do solo, pela eliminação da vegetação nativa e pela deposição irregular de resíduos sólidos.

O Brasil, país marcado pela grande desigualdade social, apresenta elementos sociais que interferem diretamente na ocorrência de enchentes. Nesse contexto, destaca-se a ocupação humana desordenada dos vales fluviais, o descarte incorreto de lixo em zonas naturais, a canalização de diversos cursos de água, além da ausência de políticas públicas de monitoramento ambiental e planejamento urbano.

Ademais, há a questão ambiental em contexto macro. Com base nas mudanças climáticas vivenciadas em escala global, espera-se que o fenômeno das enchentes seja cada vez mais potencializado em nível regional.

O território brasileiro, em comunhão com seus aspectos naturais e sociais, é extremamente vulnerável à ocorrência de enchentes, que devem tornar-se cada vez mais recorrentes e fortes nos próximos anos. Sendo assim, tal fenômeno no país deve gerar prejuízos humanos e econômicos cada vez maiores.

Um levantamento do Instituto Brasileiro de Geografia e Estatística (IBGE) constatou que 40,9% dos municípios brasileiros sofreram, nos últimos cinco anos, pelo menos um desastre natural. Foram 2.276 cidades atingidas por inundações graduais, enxurradas bruscas e/ou deslizamentos de encostas entre 2008 e 2013. O Perfil dos Municípios Brasileiros 2013 (Munic, 2013), lançado nesta quarta-feira (30), mostra que só as enchentes graduais deixaram 1.406.713 pessoas desabrigadas (definitivamente

sem casa) ou desalojadas (temporariamente sem moradia). A pesquisa constatou que 48% das 5.570 prefeituras do País não tinham instrumento para enfrentar essas ocorrências.

A maior concentração de alagamentos aconteceu nas regiões Sudeste (45,2%) e Sul (43,5%) e a menor no Centro-Oeste (19%). No Sudeste, Rio de Janeiro (88,0%) e Espírito Santo (71,8%) registraram os maiores percentuais. No Sul, o maior percentual ocorreu em Santa Catarina (60,3%). Em todo o país, 97,4% dos municípios com mais de 500 mil habitantes tiveram alagamentos.

A IoT é um importante fator estratégico em qualquer negócio, já que possibilita mudanças na rotina de trabalho. Conseqüentemente, amplia a automação dos processos, bem como o volume das análises de dados.

Logo, o primeiro benefício da internet das coisas para uma empresa é que **os** erros operacionais são menos freqüentes, logo, os custos serão reduzidos. A gestão também é beneficiada com a IoT, pois pode construir base de dados mais ampla, favorecendo as decisões estratégicas.

Inclusive, podem alterar rotinas de trabalho ao avaliar a influência de determinados fatores internos nos resultados. Em suma, a internet das coisas pode, substancialmente, trazer resultados bastante favoráveis para uma empresa.

A internet das coisas possibilita ao gestor a compreensão dos processos e, conseqüentemente, a visão geral das atividades. Com a IoT, o andamento de cada projeto pode ser acompanhado em tempo real, com problemas e necessidades.

Uma das maiores conseqüências da automatização é a redução das intervenções humanas, aumentando a produtividade. Isso acontece porque os comandos do sistema permitem às máquinas trabalhar de forma ininterrupta.

Exemplo prático é o uso de roupas inteligentes pela Ford de Valência, que permite rastrear o movimento dos funcionários, tornando as operações mais seguras e eficientes.

Entre os benefícios do IOT podemos citar:

- **Segurança da Informação**

Com a IoT, as empresas contam com infraestruturas de TI poderosas, monitoramento freqüente, garantindo a proteção das informações e dos processos.

- **Eficiência de gestão**

O uso de dispositivos de IoT possibilita trabalhar com maior volume de dados, ao mesmo tempo em que facilita sua análise. Assim, a gestão consegue promover estudos mais profundos dos negócios, identificando gargalos, modificando e aprimorando processos.

- **Redução de custos**

Ainda falando sobre eficiência de gestão, dentro dos benefícios da internet das coisas a sua empresa poderá aproveitar as reduções de custo nas operações, bem como o tempo de inatividade. Também, há a diminuição do desperdício na cadeia, uma vez que a automação otimiza o processo de produção.

- **Manutenção proativa**

Com os sensores inteligentes, a empresa monitora os aplicativos internos de forma que, com os dados específicos, a equipe de TI detecta, com antecedência, a possibilidade de alguma falha e toma medidas para evitar que ela aconteça.

3. FUNDAMENTOS DA COMUNICAÇÃO DE DADOS EM REDE

3.1 História da Internet

Com o avanço tecnológico e o barateamento de computadores pessoais e componentes eletrônicos, o computador se tornou item essencial em ambientes domésticos e corporativos. Foi aí que houve a necessidade de criação de uma rede interconectada capaz de enviar e receber informações.

Seu desenvolvimento se iniciou lá atrás, em 1969, quando a ARPANET (*Advanced Research Projects Agency Network*) se tornou o primeiro computador conectado a uma rede, desenvolvido pela ARPA (*Advanced Research Projects Agency*), um empresa controlada pelo departamento de defesa dos Estados Unidos (DOD – *Department of Defence*) e que tinha como objetivo o desenvolvimento de uma rede capaz de manter comunicação com os Estados Unidos e com a antiga União Soviética (URSS) caso houvesse uma escalada nuclear, evento esse ocorrido durante a guerra fria no período de 1945 a 1989

Essa rede revolucionou os meios de comunicação ao fazer o uso da troca de pacotes invés da conexão direta bastante usada na época. Os dados eram formatados em pacotes com um endereço de destino e então enviados pela rede e coletados pela próxima máquina. Assim a informação chegaria ao correto destino, mesmo se não houvesse uma conexão direta entre as máquinas.

Em 1990, ocorreram dois fatores que mudaram o rumo das redes de computadores e introduziram a comercialização da internet. A primeira foi a falência da ARPANET, a progenitora da internet. A MILNET (*Military Network*) foi criada e com isso boa parte dos dados de defesa foram migrados para a NSFNET. A segunda, foi a criação do primeiro ISP (*Internet Service Provider*) ou Provedor de Serviço de Internet o The World ([http:// www.world.std.com](http://www.world.std.com)), o primeiro provedor do mundo. Após isso grande parte da rede militar ao qual utilizava a NSFNET foi migrada para serviços de provedores de internet. Graças a esses eventos a internet foi introduzida em milhões de casas e empresas ao redor do mundo. A internet também serviu como uma base para vários outros serviços e aplicações.

3.2 Redes de Computadores

Podemos definir uma rede de computadores como: alguns computadores conectados entre si através de algum meio de comunicação (cabos ou sem fio), todos com o propósito de compartilhar e trocar informações um com o outro. Existem também dispositivos que servem como intermediador de comunicação entre dois dispositivos conhecidos como dispositivos de rede, entre os mais comuns podemos encontrar:

- **Roteador:** opera na camada de rede do modelo OSI, com a ajuda de algoritmos de rota, consegue escolher a melhor rota de entrega.
- **Hub:** um dispositivo simples e barato se comparado a outros dispositivos, opera sobre a camada física do modelo OSI e conecta vários computadores a uma LAN (*Local Area Network*). É considerado menos inteligente pelo fato de não conseguir filtrar dados e não saber o destino do pacote, todos os pacotes são enviados a todos os dispositivos da rede
- **Switch:** um dispositivo que opera sobre a camada de dados do modelo OSI. Ele recebe pacotes dos dispositivos conectados fisicamente as portas e os envia novamente, porém somente através das portas que os pacotes devem chegar. Através do endereço MAC (*Media Access Control*) o switch pode identificar o remetente e o destinatário do pacote. Também podem operar na 3 camada alterando rotas.

Esses dispositivos não se limitam apenas a criação de redes locais (LAN), as redes de computadores podem ser mais amplas do que imaginamos. Elas são categorizadas levando em consideração seu tamanho e a quantidade de dispositivos conectados a ela. Entre os tipos de redes existentes podemos citar:

- **LAN (*Local Area Network*):** essas redes conectam dispositivos presentes dentro de um mesmo espaço limitado como casas ou pequenos negócios.
- **WLAN (*Wireless Local Area Network*):** redes semelhantes a LAN, porém utilizam sistemas sem fio para a troca de informações.
- **MAN (*Metropolitan Area Network*):** possui um alcance maior que a LAN, pode abranger cidades, estados ou regiões metropolitanas. Ou por exemplo empresas que precisam interligar filiais na mesma cidade.
- **WAN (*Wide Area Network*):** esse tipo de conexão pode abranger países e até mesmo continentes.

3.2 Modelos, Protocolos e Serviços de rede

Toda comunicação entre dois dispositivos requer um conjunto de regras que devem ser seguidas, a fim de que a conexão de fato seja funcional. Essas regras determinam os padrões do endereçamento, os níveis de segurança, endereçamento etc., e são agrupadas em camadas de acordo com sua finalidade formando assim um modelo a ser seguido. Existem hoje dois modelos utilizados, o modelo OSI e o modelo TCP/IP.

3.2.1 Modelo OSI (*Open Systems Interconnection*)

O modelo OSI organiza a rede em sete camadas. Essas camadas definem tanto o hardware como software de rede manipulam a informação e a transferência através da rede. Interoperabilidade, o proposito para definir um padrão de protocolo existe quando há compatibilidade entre os protocolos dos dispositivos. Cada camada é capaz de se comunicar com a camada correspondente do receptor.

Figura 1- Modelo OSI



Fonte: https://www.alura.com.br/artigos/assets/uploads/2017/12/image_0.jpg

O modelo possui 7 camadas:

- **Camada 1 – Física:** A camada física é a camada mais baixa do modelo, ela abrange a parte da rede responsável pela transferência e recepção dos bits. Nessa camada é especificado os dispositivos utilizados como meio de comunicação, os dados são então processados e enviados a próxima camada.
- **Camada 2 – Enlace:** Essa camada observa se algum pacote possui defeito e controla o fluxo de envio dos pacotes.

- **Camada 3 – Rede:** Nessa camada se encontra o endereço de IP de origem e destino, nessa camada pode ser decidido a rota que será seguida para a entrega do pacote.
- **Camada 4 – Transporte:** Essa camada garante que os pacotes vindos da camada 3 cheguem ao seu destino. Ela gerencia o transporte dos pacotes para garantir o sucesso no envio e recebimento de dados.
- **Camada 5 – Sessão:** Essa camada é responsável por estabelecer e terminar a conexão entre hosts, ela também inicia e sincroniza os hosts, registros de logs e tarefas de segurança.
- **Camada 6 – Apresentação:** Aqui ocorre a tradução dos dados para que o próximo a utilize. Essa camada é a responsável pela conversão de caracteres, conversão, compactação e criptografia de dados.
- **Camada 7 – Aplicação:** Aqui ocorre o consumo dos dados, a interação entre o usuário e a máquina.

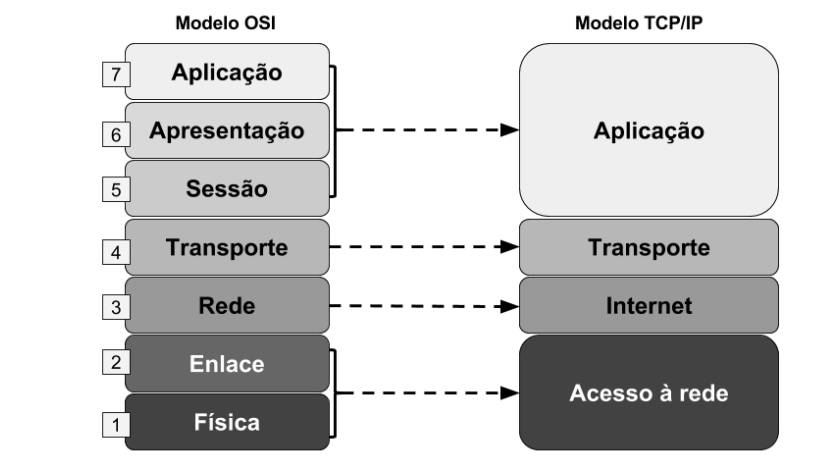
3.2.2 Modelo TCP/IP

O modelo TCP/IP (*Transfer Control Protocol*) se parece muito com o modelo OSI, os dois são vastamente usados na comunicação de protocolos de rede. A principal diferença entre eles é que o modelo OSI é um modelo conceitual e praticamente não é usado para comunicação. Ao invés disso, define como as aplicações podem se comunicar na rede. Já o modelo TCP/IP, por outro lado, é vastamente usado para se estabelecer links interação na rede.

O protocolo TCP/IP se baseia nos padrões de criação da internet, enquanto o modelo OSI prove direção no modo em que a comunicação deve ser realizada. Por isso o modelo TCP/IP se tornou o modelo mais prático.

Os modelos TCP/IP e OSI, que embora parecidos, possuem algumas diferenças. A principal semelhança está no modo em que ambas foram construídas, ambas usam o conceito de camadas, embora o modelo TCP/IP consiste em apenas quatro camadas, enquanto seu antecessor, o modelo OSI, possui sete camadas.

Figura 2 - Modelo OSI e TCP/IP



Fonte: <https://www.researchgate.net/AS:428418383781889@1479154310501/Figura-3-Modelo-OSI-e-TCP-IP.png>

Entre as camadas do modelo TCP/IP podemos encontrar:

- **Camada 1 – Acesso a Rede ou Datalink:** define como os dados devem ser enviados e recebidos, é o responsável por transmitir informação entre as aplicações e dispositivos na rede através de cabos de redes, placas de internet, ou dispositivos sem fio, engloba tanto a camada Física como a de Enlace.
- **Camada 2 – Internet:** a camada responsável por enviar pacotes pela rede e controlar seu movimento para assegurar que eles cheguem ao seu destino. Prove as funções e procedimentos para a transferência de dados entre aplicações e dispositivos pela rede
- **Camada 3 – Transporte:** responsável por manter uma conexão sólida e estável entre a aplicação ou dispositivo de origem e seu destino. Nessa camada os dados são divididos em pacotes e numerados para se criar uma seqüência. A camada de transporte logo em seguida determina o tamanho dos dados a serem enviados, para onde serão enviados e a que taxa. Essa camada se assegura que os pacotes de dados serão enviados sem erros e em uma seqüência correta, logo em seguida recebe confirmação de que os dados foram entregues ao destinatário.
- **Camada 4 – Aplicação:** a camada de aplicação se refere aos programas que fazem o uso do protocolo TCP/IP para se comunicarem entre si. Essa é a camada que os usuários geralmente interagem, enviando e-mails, acessando

a internet ou transferindo arquivos. Ela combina as camadas de sessão, apresentação e aplicação do modelo OSI.

3.2.3 Protocolos de Rede

Um protocolo de rede é um conjunto de regras que dizem como formatar, transmitir e receber pacotes de dados para que computadores, servidores, roteadores e dispositivos possam se comunicar independente da estrutura, design ou padrões.

Para conseguir enviar e receber informações, os dispositivos nas duas pontas de comunicação devem aceitar e seguir os padrões do protocolo, na rede esses protocolos podem ser construídos dentro dos softwares, hardwares ou em ambos.

Sem esses protocolos os computadores e dispositivos não sabem como se comunicar. Como resultado, exceto em redes com protocolos construídos em torno de uma arquitetura específica ou proprietária, algumas redes funcionariam, e a internet ao qual conhecemos jamais existiria. Os protocolos mais usados hoje em dia são os do modelo TCP/IP, bastante usado em modelos cliente-servidor, dentro da camada de transporte podemos encontrar:

- **TCP (*Transmission Control Protocol*)**, que usa um conjunto de regras para a troca de mensagens o que garante a confiabilidade, uma conexão é estabelecida antes da transferência de arquivos começar. Os dados são enviados sem erros ou duplicação e é recebido na mesma ordem que é enviado, esse protocolo foca na segurança dos dados entregues e trata os dados como um conjunto de bytes.
- **UDP (*User Datagram Protocol*)**, uma alternativa ao protocolo TCP, prove uma conexão não segura e instável entre as aplicações, os dados são transmitidos link a link, não há conexões ponto a ponto. O protocolo também não garante que os pacotes sejam entregues, nesses casos os pacotes podem ser perdidos ou duplicados e os datagramas podem ser recebidos em ordem incorreta. Seu uso é focado em velocidade, como jogos de computadores e sistemas em tempo real, onde a perda de pacotes pode ser irrelevante.
- **IP (*Internet Protocol*)**, um protocolo a nível de rede. Ele prove um datagrama de serviços entre as aplicações, dando suporte a ambos TCP e UDP.

Dentro da camada de aplicação podemos encontrar alguns protocolos bastante conhecidos e utilizados por usuários:

- **DHCP (*Dynamic Host configuration Protocol*)**, um protocolo que permite que administradores de rede e usuários automatizem o processo de entrega de endereços IP em uma rede.
- **DNS (*Domain Name Service*)**, ajuda na tradução de endereços IP, como memorizá-los se torna algo complicado, o DNS traduz um nome comum para um endereço IP e redireciona a aplicação para ele;
- **FTP (*File Transfer Protocol*)**, permite o envio e recebimento de arquivos entre os computadores e é gerenciado pelo protocolo TCP. Usa duas conexões TCP, uma para controle e outra para conexão, a conexão de controle serve para controlar informações como senhas, comandos e armazenar arquivos, o controle de dados serve para transferir o arquivo em questão. Ambas as conexões rodam em paralelo.
- **HTTP (*Hyper Text Transfer Protocol*)**, trabalha em um modelo de cliente-servidor, onde navegador web age como um cliente. Arquivos como texto, imagens e outros arquivos de mídia são compartilhados na internet usando o protocolo HTTP. Como um protocolo de requisição e resposta, o cliente envia uma requisição ao servidor, que é processada e enviado uma resposta ao cliente. O protocolo não possui estado, o cliente e servidor não se conhecem enquanto a conexão é intacta. Após a requisição o servidor esquece a existência do cliente.

4.DESENVOLVIMENTO

Na fase inicial do desenvolvimento foi discutido a questão das enchentes aos quais todos os anos causam prejuízos e mortes em algumas partes do Brasil e as crescentes invasões de água as ruas e áreas de encostas situadas ao redor de mares e rios.

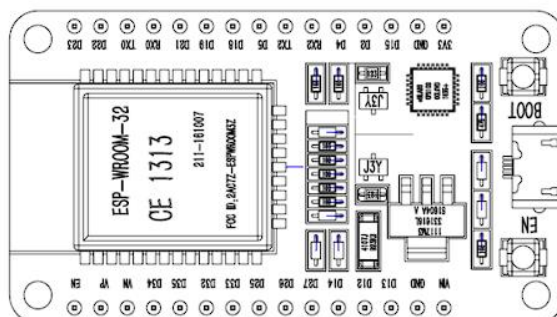
A partir de ideais e coletas de informações, foi decidido a criação de um dispositivo capaz de medir a distância entre o solo e o sensor e partir daí presumir uma possível enchente ou nível de água incomum. Esse dispositivo pode auxiliar no monitoramento do nível de água bem como na coleta de dados que podem ser acessadas por outros dispositivos conectados em rede.

4.1 Parte Física (Hardware)

Existem uma infinidade de dispositivos que podem ser usados como sistemas IOT, no entanto alguns deles apresentam valores elevados um caro custo de implementação.

Para o projeto em questão foi decidido o uso de um dispositivo que seja relativamente barato e de fácil implementação. O ESP32 surgiu como uma opção viável, se comparado ao seu concorrente o Arduino, ao qual, apesar do baixo custo, se torna inviável já que há uma necessidade de obter módulos externos para comunicação de dados, o que torna o processo de desenvolvimento confuso e demorado.

Figura 3 - ESP32



Fonte:

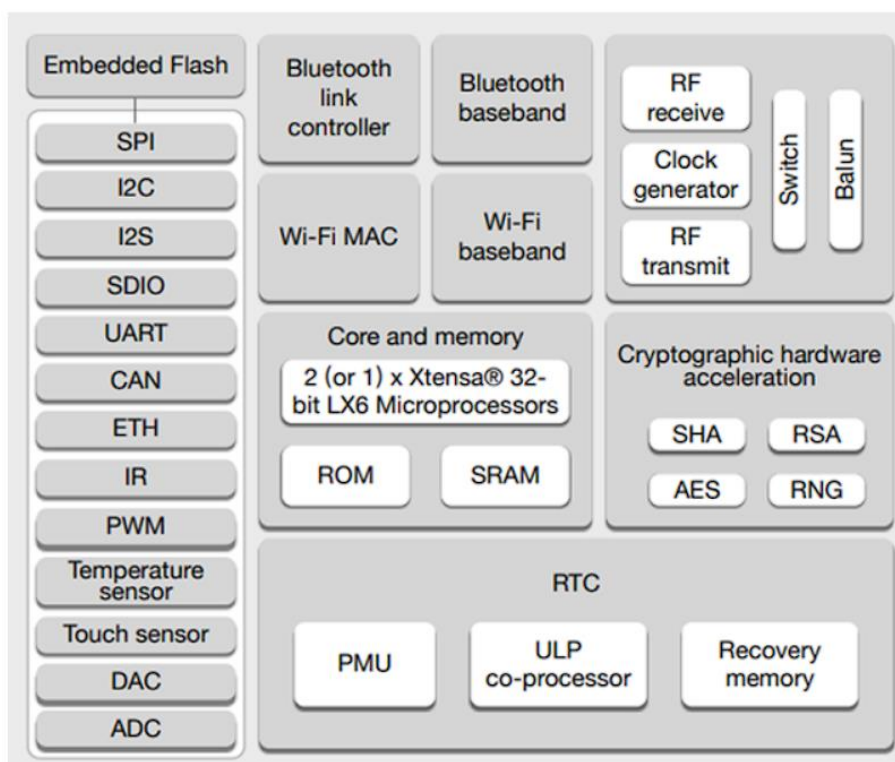
<https://d2t1xqejof9utc.cloudfront.net/screenshots/pics/760b9dff1897a530995e1fc53764f459/large.PNG>

Apesar de seu tamanho reduzido, se comparado ao Arduino, o Esp32 apresenta as seguintes características:

- **Conexão de Rede sem fio e Bluetooth:** possibilidade de conexão sem fio 802.11 b/g/n com dispositivos usando protocolos de redes e conexão serial sobre o protocolo serial *Bluetooth* v4.2 BR/EDR e BLE.
- **Baixo consumo de energia:** o dispositivo consome uma baixa corrente de energia elétrica, o que torna possível sua utilização junto com baterias ou sistemas de recarga solar, reduzindo impactos ao meio ambiente.
- **Processador Dual Core:** faz uso de um processador Xtensa *dual-core* de 32 bits que opera em 240Mhz, o que torna o uso de *Multithreads* uma realidade, é possível realizar operações paralelas sem a latência ou gargalo na entrega de informações.

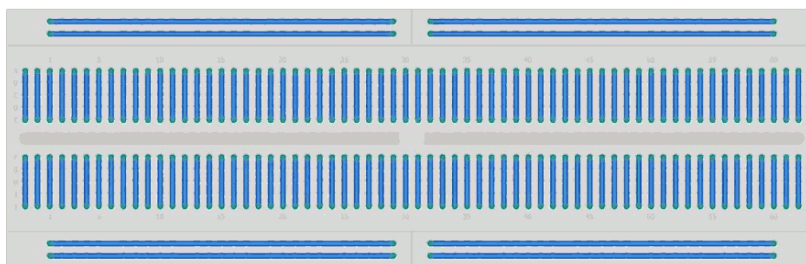
Devido a sua arquitetura e desempenho o dispositivo se mostrou excelente tanto na parte de programação do dispositivo que será vista logo em seguida como na parte de testes, onde foi possível alimentar ambos os dispositivos, sensor e leds com apenas uma porta USB.

Figura 4 - Arquitetura ESP32



O dispositivo foi montado em uma protoboard para facilitar o uso, *protoboards* são pequenas placas com furos que se interligam para criar circuitos, esse processo se torna ótimo na criação de protótipos, já que não há a necessidade de se obter materiais caros como estações de ar quente ou estações de solda, o que em muitos casos dobraria o orçamento do projeto.

Figura 5 - Protoboard



Fonte: <https://www.robocore.net/tutoriais/como-utilizar-uma-protoboard>

Para a interligação dos pontos foram utilizados alguns conectores (macho x macho) para a *protoboard* e (macho x fêmea) para o dispositivo.

Figura 6 - Cabos de conexão da protoboard



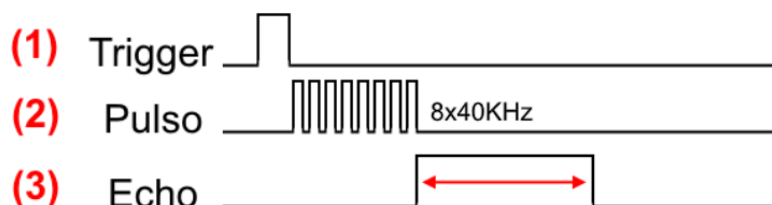
Fonte: <https://compeljundiai.com.br/produto/cabo-jumper-macho-x-macho-20cm-40-vias/>

Foram utilizados três leds verdes para a indicação de atividade da placa, além de resistores para impedir que os leds queimem.

O sensor utilizado foi um HC-SR04, o elemento fundamental do projeto, por se tratar de um protótipo esse sensor de baixo custo se tornou uma opção vantajosa se comparado a um sensor profissional que custa o triplo de seu valor. Esse sensor pode realizar leituras de distancias entre 2cm e 4 metros com uma precisão de 3mm e seu funcionamento se baseia no envio de sinais ultrasônicos que são recebidos (*Echo*) e com base no tempo do envio e da

recepção desse sinal, calculamos a distância entre o sensor e o objeto utilizando a equação (distância = (Tempo *echo* nível alto * velocidade do som) / 2). O sensor é conectado utilizando quatro pinos: *Vcc* (5v corrente contínua), *Trigger*, *Echo* e GND(*Ground*).

Figura 7 - Funcionamento do sensor ultrassônico

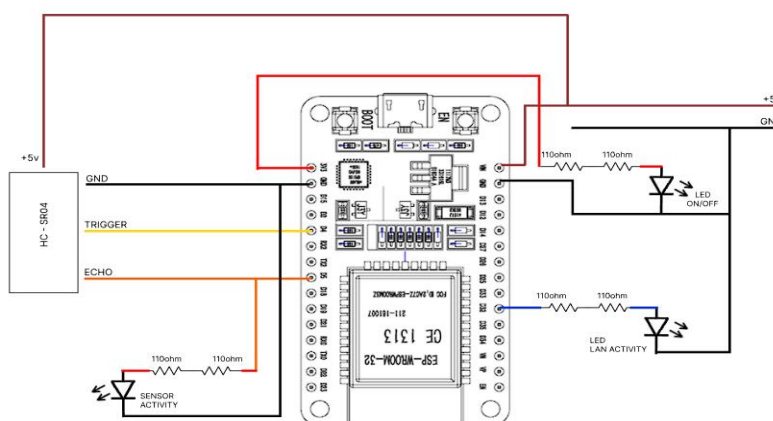


Fonte: <https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino/>

Utilizando os componentes certos e através de muito esforço e aprendizado foi possível a criação de um dispositivo confiável e que funciona como pretendido, utilizando um baixo consumo e entregando uma performance ótima dentro dos padrões previstos.

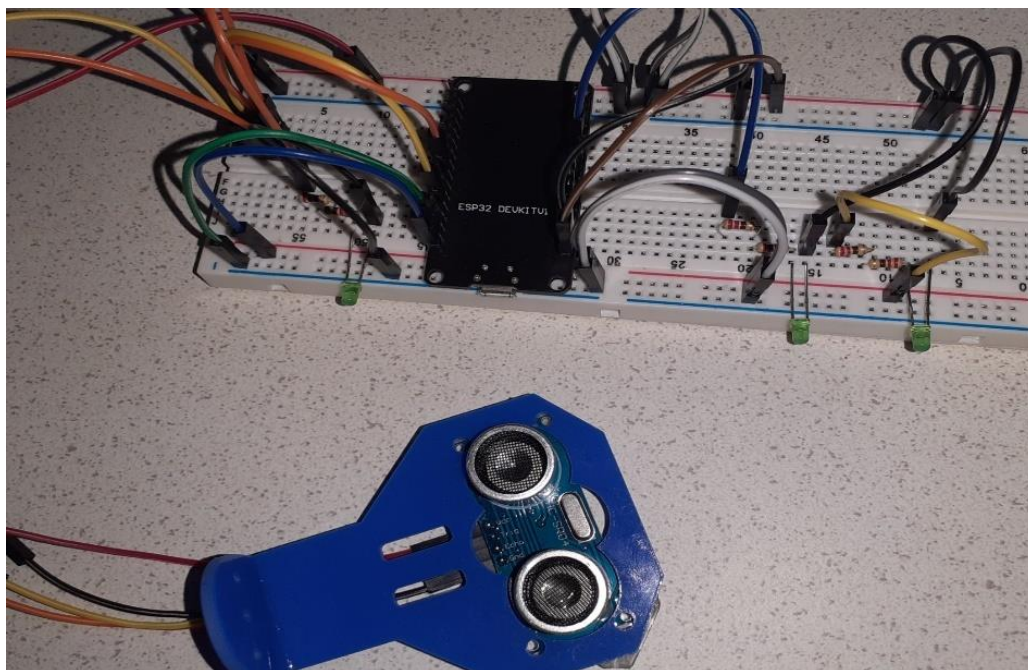
Logo abaixo é possível encontrar um esquema elétrico do projeto, suas ligações e componentes:

Figura 8 - Esquema elétrico do dispositivo



Fonte: Autoria Própria

Figura 9 - Dispositivo montado



Fonte: Autoria Própria

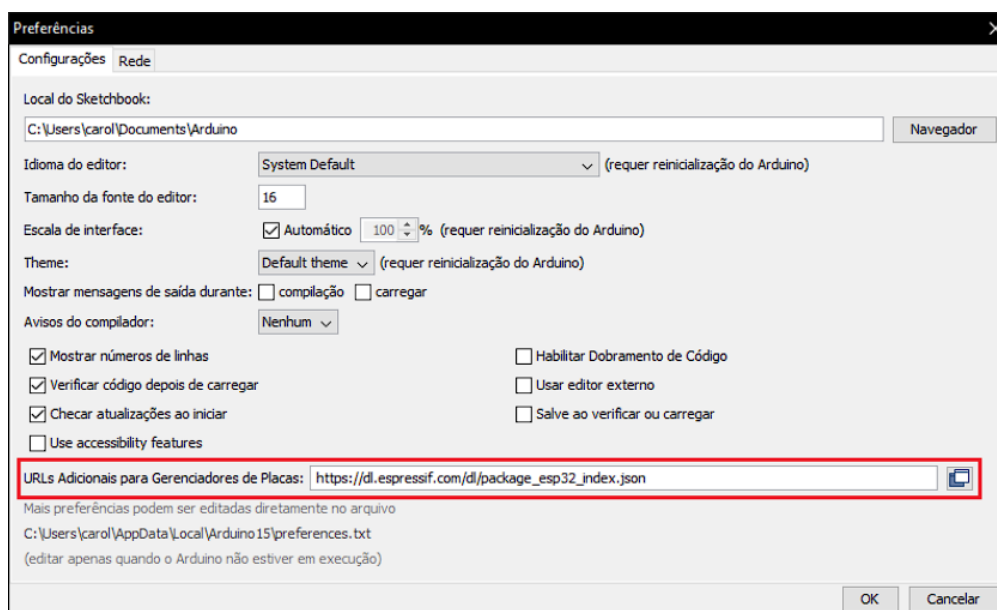
4.2 Parte Lógica

4.2.1 Desenvolvimento do Firmware do Dispositivo (Software)

O desenvolvimento do firmware foi feito utilizando ambos, compilador e IDE (*Integrated Development Environment*) do Arduino que pode ser obtido através de endereço: <https://www.arduino.cc/en/software>. Ambos são leves e rodam em sistemas variados como Windows, Linux e Mac OS, a única necessidade é a instalação do *Java Runtime Environment*, já que a linguagem foi utilizada para a criação da IDE, além do interpretador Python para o processo de transferência do firmware para o dispositivo já que o Arduino não dá suporte ao ESP32.

O primeiro passo foi a instalação de uma biblioteca com funções que são utilizadas no ESP32, como dito anteriormente, o dispositivo não apresenta suporte com a IDE sendo necessário a instalação de frameworks e arquivos de terceiros através do link https://dl.espressif.com/dl/package_esp32_index.json.

Figura 10 - Instalação de bibliotecas adicionais



Fonte: <https://www.blogdarobotica.com/2021/08/24/como-programar-a-placa-esp32-no-arduino-ide/>

Com a placa devidamente instalada, foi possível dar início ao desenvolvimento. Para a programação do dispositivo foi utilizada a linguagem C/C++, padrões da IDE do Arduino, embora seja possível programar em outras linguagens como Python ou até mesmo Javascript. No entanto o tamanho do código se tornaria um pouco extenso pela quantidade de bibliotecas utilizadas por outras linguagens, o que para dispositivos com memória limitada se tornaria perigoso já que não se consegue prever quanto espaço irá ocupar.

O primeiro passo foi idealizar o que o dispositivo iria fazer, primeiramente ele coletaria os dados do sensor e os armazenaria em uma estrutura de dados, logo após geraria um arquivo JSON (*JavaScript Object Notation*). Logo depois esse arquivo seria enviado ao cliente em caso de uma requisição.

No entanto a criação de um servidor REST em um dispositivo com menos de 64KB de memória seria algo um tanto quanto complicado, já que a cada requisição o dispositivo demorava cerca de 30ms para a entrega dos dados, isso gerava gargalos na rede e falta de comunicação com o dispositivo, fazendo com que um sistema em tempo real apresentasse atrasos.

Depois de pesquisar a documentação do dispositivo foi possível compreender que ele oferecia a possibilidade de execução de processos em paralelo (*Multithread*) já que ele oferece suporte a RTOS (*Real Time Operating Systems*), onde o tempo de

resposta é mais importante do que executar várias tarefas ao mesmo tempo. Foram criadas duas funções, uma para executar a leitura do sensor e outra para atender as requisições HTTP do cliente, foi definido também um manipulador (*handler*) para informar o sistema sobre qual thread está operando e em qual núcleo.

Figura 11 - Funções Principais

```
5 // Cria um task handler que ira gerenciar os estados das tarefas
6 // E as funções que serao passadas ao respectivos cores
7 TaskHandle_t Task1;
8 void TaskDaemon(void *pvParameters);
9 void TaskHandleConnectionLed(void *pvParameters);
```

Fonte: Autoria Própria

Logo após travamos as funções em seus respectivos núcleos:

Figura 12 – Funções Multithread

```
/* Trava as duas funções em nucleos de processamento diferentes.
   O nucleo 0 ira gerenciar a rotina do sensor
   O nucleo 1 ira gerenciar as rotinas de conexão e rotas da api
*/
xTaskCreatePinnedToCore(TaskDaemon, "HandleDaemon", 10000, NULL, 1, NULL, 0);
xTaskCreatePinnedToCore(TaskHandleConnectionLed, "HandleConnectionLed", 10000, NULL, 2, NULL, 1);
}
```

Fonte: Autoria própria

Ambas as funções TaskDaemon e TaskHandleConnectionLed são gerenciadas pelo sistema RTOS e não mais pela função loop do arquivo principal, as funções contém um loop infinito que os executara até que recebam um sinal de parada.

No arquivo Server.cpp se encontra a rotina principal do sistema, aqui são definidas as funções do sensor e as funções de gerenciamento de rotas e clientes que acessam o dispositivo usando o protocolo HTTP. Primeiro definimos um objeto que irá acessar as funções do sensor a atribuímos os pinos do ESP32 a ele, logo após definimos um objeto Webserver e a nossa estrutura de dados que irá conter as informações do sensor atual, ou mais de um. A função “sensorDaemon” faz a leitura do sensor e logo após gera um arquivo JSON que mais tarde poderá ser enviado ao apresentado ao console em forma de informação de depuração.

A função “setDevice” recebe os dados documento JSON que serão enviados via método post ao servidor, caso a informação não esteja corretamente formatada ou se os dados inseridos forem inválidos o processamento é interrompido e uma mensagem de erro será gerado como uma resposta JSON. Caso as informações

inseridas sejam validas serão atribuídas a estrutura da dados, ao cliente será enviado uma resposta de sucesso (200) em forma de um arquivo JSON, que caso necessite pode ser usado na interface web. Como medida de segurança apenas o nome do dispositivo e sua distância de segurança podem ser alteradas, a fim de evitar o acionamento indevido do alarme.

Figura 13 - Manipulação de objetos JSON

```

74  if (server.method() == HTTP_POST)
75  {
76      if (postObject.containsKey("device") && postObject.containsKey("safeDistance"))
77      {
78          String buf;
79
80          sensor.deviceName = document["device"].as<String>();
81          sensor.safeDistance = document["safeDistance"];
82          sensor.alert = (calculateDistance() < sensor.safeDistance) ? 1 : 0;
83
84          serializeJson(document, buf);
85          server.send(201, "application/json", buf);
86          getClientStatus("/api/v1/reportaenchente/settings/setdevice");
87          buf.clear();
88      }
89  }else{
90      document["Erro"] = "Formato invalido";
91      serializeJson(document, buf);
92      server.send(400, "application/json", buf);
93  }
94 }
```

Fonte: Autoria Própria

Para calcular a distância correta entre o solo e o sensor utilizamos a função “calculateDistance” que logo após um pulso do sensor, convertera o tempo decorrido em microssegundos em centímetros por segundo, no entanto a função só é executada se o sensor estiver online.

Figura 14 - Cálculo de distância

```

100 float calculateDistance()
101 {
102     float cmMsec = 0;
103     if (sensor.status == "online")
104     {
105         long microsec = ultrasonic.timing();
106         cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
107         return cmMsec;
108     }
109     return cmMsec;
110 }
```

Fonte: Autoria própria

As rotas são definidas usando a função “setRoutes”, aqui é importante habilitar o CORS (*Cross-Origin Resource Sharing*) para permitir o redirecionamento entre hosts. A rota principal do servidor REST pode ser encontrada através do endereço `http://192.168.0.5:3333/api/v1/reportaenchente/`, a partir daí existem duas rotas:

- “/stats”: gera um arquivo JSON com as estatísticas do dispositivo.
- “/setdevice”: aqui é possível alterar algumas informações do sensor.

Figura 15 - Rotas

```

145 // Cria as rotas de acesso ao servidor REST
146 void setRoutes()
147 {
148     server.enableCORS();
149     server.on("/api/v1/reportaenchente/stats", getStatus);
150     server.on("/api/v1/reportaenchente/setdevice", HTTP_POST, setDevice);
151     server.onNotFound(notFound);
152     server.begin();
153 }
```

Fonte: Autoria própria

No arquivo `WifiSettings.cpp`, será utilizado para realizar a conexão do dispositivo com a internet sem fio, o sistema tenta se autenticar e em caso de sucesso exibe uma informação de sucesso incluindo o endereço IP, máscara de rede e gateway da rede. Em casos em que não seja possível se autenticar, a função se repetirá até uma conexão seja estabelecida.

Figura 16 - Função de conexão

```

8 void connectToNetwork() {
9     Serial.print("Conectando-se a rede ");
10    Serial.print(DEFAULT_SSID_NAME);
11
12
13    WiFi.begin(DEFAULT_SSID_NAME, DEFAULT_SSID_KEY);
14    while(WiFi.status() != WL_CONNECTED) {
15        Serial.print(".");
16        delay(1000);
17    }
18
19    Serial.println("");
20    Serial.println("Conectado com sucesso!");
21    displayConnectionStatus();
22
23 }
```

Fonte: Autoria própria

Já o arquivo “handleJson.cpp” como o nome sugere, manipulava os arquivos JSON, aqui eles serão criados com base nas informações passadas pelo dispositivo.

Figura 17 - Função de criação do objeto JSON

```

4
5 // Cria o objeto JSON que sera enviado ao cliente baseado nos parametros do sensor
6 String createDeviceJson(String device, String status, float value, float safeDistance,
7   String measureUnit, StaticJsonDocument<250>& jsonObject){
8   jsonObject.clear();
9   jsonObject["device"] = device;
10  jsonObject["status"] = status;
11  jsonObject["distance"] = value;
12  jsonObject["safeDistance"] = safeDistance;
13  jsonObject["measureUnit"] = measureUnit;
14  jsonObject["alert"] = (value < safeDistance) ? 1 : 0;
15  serializeJson(jsonObject, buffer);
16  return buffer;
17 }

```

Fonte: Autoria própria

Logo após o processo de codificação e instalação da imagem no dispositivo (Flash), com ajuda do monitor serial do Arduino é possível ver o que acontece no dispositivo.

Figura 18 - Funcionamento do dispositivo

```

19:22:37.306 -> Conectado com sucesso!
19:22:37.306 -> =====
19:22:37.306 -> IP: 10.0.0.2
19:22:37.340 -> NetMask: 255.0.0.0
19:22:37.340 -> Gateway: 10.0.0.1
19:22:37.340 -> =====
19:22:37.374 -> [DEBUG]Sensor -> {"device":"device1","status":"online","distance":204.4831696,"safeDistance":30,"measureUnit":"cm","alert":0}
19:22:38.366 -> [DEBUG]Sensor -> {"device":"device1","status":"online","distance":204.5193634,"safeDistance":30,"measureUnit":"cm","alert":0}
19:22:39.426 -> [DEBUG]Sensor -> {"device":"device1","status":"online","distance":204.5374756,"safeDistance":30,"measureUnit":"cm","alert":0}
19:22:40.481 -> [DEBUG]Sensor -> {"device":"device1","status":"online","distance":204.9537811,"safeDistance":30,"measureUnit":"cm","alert":0}
19:22:41.435 -> [DEBUG]Sensor -> {"device":"device1","status":"online","distance":204.4831696,"safeDistance":30,"measureUnit":"cm","alert":0}
19:22:42.427 -> [DEBUG]Sensor -> {"device":"device1","status":"online","distance":204.4831696,"safeDistance":30,"measureUnit":"cm","alert":0}

```

Fonte: Autoria própria

4.2.2 Software de visualização dos dados

Após a criação do dispositivo, era necessária uma ferramenta que auxiliasse o processo de interpretação dos dados gerados pelo sensor. Entre os requisitos estava a necessidade de um sistema de monitoramento em tempo real.

Com a ajuda do sistema RTOS do ESP32 essa tarefa se tornou mais fácil já que o servidor REST pode responder mais de duas mil requisições por segundo, como os arquivos gerados consomem menos de 1KB de armazenamento, o dispositivo não sofre nenhum tipo de anomalia ao enviar e receber os dados.

Para visualizar os dados utilizamos um minissistema web desenvolvido utilizando a linguagem Python (<https://www.python.org>) e o *framework* Flask (<https://flask.palletsprojects.com/>), nada muito complexo foi desenvolvido aqui, apenas um mecanismo que permite visualizar os dados do sensor em tempo real.

Figura 19 - Python e Flask



Fonte: <https://dev.to/lucianopereira86/python-flask-part-3-local-database-30pk>

A aplicação web faz uso de páginas HTML com interpolação de dados, onde é possível passar parâmetros e dados e as visualizar no browser. Primeiro definimos as rotas que compõe nossa aplicação, elas são definidas no arquivo “main.py”:

- “/”: visualiza as informações em tempo real do dispositivo.
- “/data”: obtém um arquivo JSON do dispositivo.
- “/parâmetros”: gera um arquivo JSON para alteração de dados via método POST.
- “/dispositivos”: exibe a lista de sensores presentes e seus dados

Figura 20 - Arquivo main.py

```

7  @app.route('/', methods={'GET'})
8  def overview():
9      return render_template('overview.htm')
10
11 @app.route('/data', methods={'GET'})
12 def parseJson():
13     data = requests.get("http://192.168.0.5:3333/api/v1/reportaenchente/stats")
14     return data.json()
15
16
17 @app.route('/parametros', methods={'GET', 'POST'})
18 def parametros():
19     if request.method == 'POST':
20         deviceName = request.form.get("sensorName")
21         distance = request.form.get("distance")
22         print(deviceName)
23
24         response = requests.post("http://192.168.0.5:3333/api/v1/reportaenchente/setdevice",
25                                 json={"device" : deviceName, "status" : "", "safeDistance" : distance, "measureUnit" : "cm" })
26         if response.ok:
27             print(response.json())
28
29         return render_template('overview.htm')
30
31     return render_template('parametros.htm')
32
33 @app.route('/dispositivos', methods={'GET'})
34 def dispositivos():
35     return render_template('dispositivos.htm')

```

Fonte: Autoria própria

A geração de gráficos se tornou mais fácil graças a utilização da biblioteca Chart.js, que foi construído para gerar gráficos das mais variadas formas e facilitar o processo de desenvolvimento.

Figura 21 - Chart.js



Fonte: <https://www.chartjs.org/>

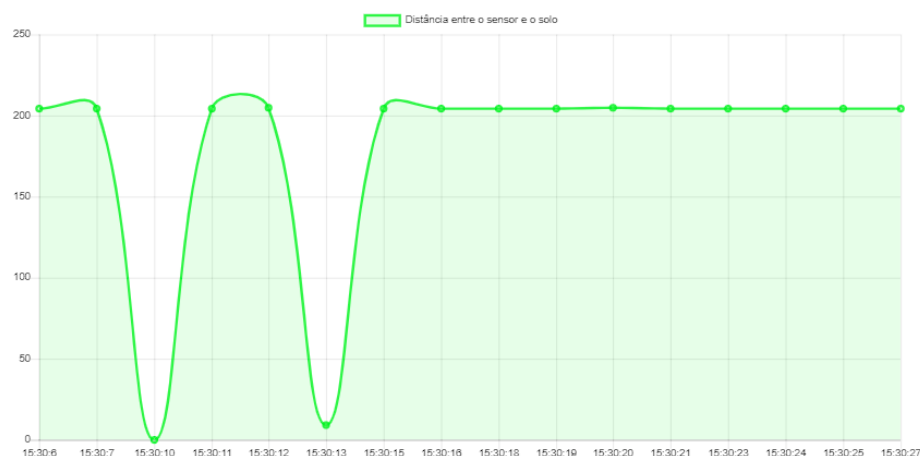
No arquivo chart_data.js, obtemos os dados do sensor através da rota “/data”, logo após com a ajuda da biblioteca JQuery(<https://jquery.com/>) passamos os dados recebidos para o gráfico que será exibido, a função “setTimeout” repetira o processo a cada um segundo criando assim uma visualização mais realista. Para manter o

gráfico o mais limpo possível, o gráfico é limpo a cada dez requisições feitas ao servidor.

Através do arquivo “weather.js” é possível obter os dados climáticos em que o sensor se encontra e com isso exibi-los no card condições climáticas.

Figura 22 - Visualização dos dados (condições normais)

MONITORAMENTO DE ENCHENTES



Distancia

Distância entre o solo e o sensor (quanto maior a distância menor o risco)



204.2 cm

Distância segura

Margem de segurança entre o sensor e o solo



30 cm

ALERTA

Se o nível estiver abaixo da distância de segurança um estado de alerta é iniciado



desligado

Condições

Condições climáticas da região ao qual o sensor se encontra



20.9°

Fonte: Autoria própria


Figura 23 - Visualização de dados (alerta de enchente)

MONITORAMENTO DE ENCHENTES



Fonte: Autoria própria

Através da rota “/parâmetros” é possível alterar os dados do sensor, se eles forem permitidos.

Figura 24 - Alterar parâmetros

The image shows a web form titled "Alterar Parâmetros". It contains two input fields: "Nome do sensor" with a placeholder "Novo nome" and "Distância segura" with a placeholder "ex: 55.0". Below the fields is a button labeled "Alterar".

Fonte: Autoria própria

A rota “/dispositivos” exibe os sensores presentes no dispositivo.

Figura 25 - Dispositivos

The image shows a web page titled "Dispositivos". It displays the following sensor data:

- Sensor: device1
- Distância: 204.4288635
- Distância Segura: 30
- Status: online
- Alerta: Desligado

Fonte: Autoria própria.

O vídeo de demonstração do projeto em funcionamento pode ser encontrado em (<https://www.youtube.com/watch?v=tcB7zN2U03A>), nele é possível compreender o funcionamento e como o sistema em tempo real interpreta os dados.

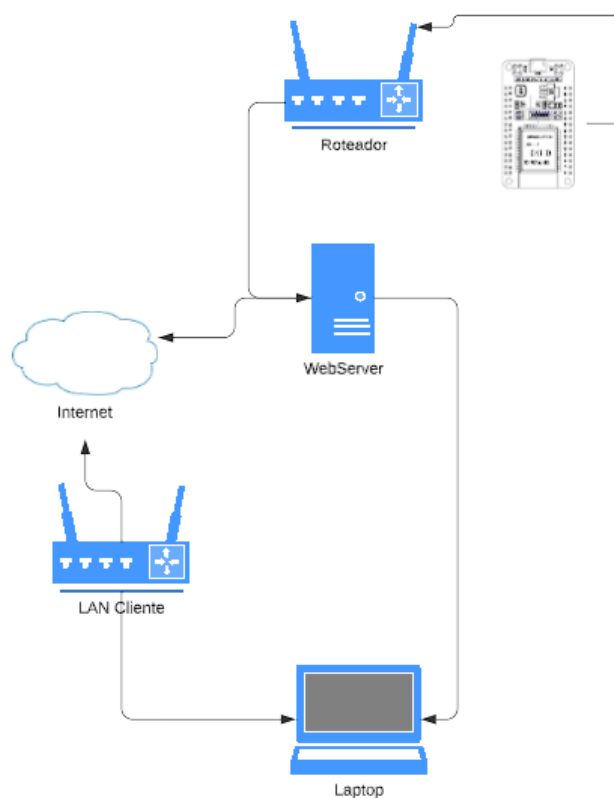
Como o código fonte do projeto contém muitas linhas de código, não é possível explicar linha por linha aqui, no entanto, ele se encontra em um repositório do GitHub para consulta (https://github.com/nunees/Trabalho_APS_5sem).

5. ESTRUTURA DO PROJETO

5.1 Diagrama Lógico

O diagrama logico da rede utilizada foi construído da seguinte forma, no topo temos um roteador ao qual nosso controlador ESP32 se conectara utilizando uma rede sem fio. Logo abaixo se encontra nosso servidor web, ele é o responsável por acessar a API do nosso controlador e interpretar os dados recebidos em forma de páginas HTML, vale ressaltar que esse Web Server pode estar conectado ou não a uma rede de internet, podendo assim, fazer com que as páginas sejam acessadas por qualquer pessoa na rede e na internet. O roteador do lado do cliente é opcional, se faz necessário caso ele não tenha acesso físico ao Web Server.

Figura 26 - Diagrama lógico



Fonte: Autoria própria

Os dispositivos foram configurados da seguinte maneira:

- **Roteador:**
 - Classe: A
 - Tipo de Conexão: Cabo par-trançado

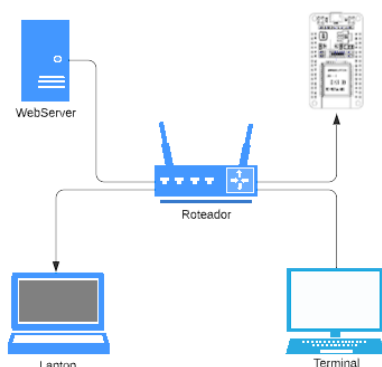
- IP estático: 10.0.0.1
 - Máscara: 255.0.0.0
 - Gateway: 10.0.0.1
- **ESP32:**
 - Classe: A
 - Tipo de Conexão: Sem fio
 - IP estático: 10.0.0.2
 - Máscara: 255.0.0.0
 - Gateway: 10.0.0.1
- **Web Server:**
 - Classe: A
 - Tipo de Conexão: Cabo par trançado ou sem fio
 - IP estático: 10.0.0.3 ou 10.0.0.4
 - Máscara: 255.0.0.0
 - Gateway: 10.0.0.1
- **Roteador Cliente (Opcional)**
 - Classe: B
 - Tipo de Conexão: Cabo par trançado ou sem fio
 - IP DHCP: 192.168.0.100/24
 - Máscara: 255.255.255.0
 - Gateway: 192.168.0.1

5.2 Topologia

O projeto utiliza a topologia estrela, esse tipo de configuração é a mais comum. A rede é organizada de modo que os nós sejam conectados a um dispositivo central que no caso irá atuar como um servidor central. Ambos os dispositivos transmitem e recebem dados em velocidade de 100mb/s para redes cabeadas e 54mb/s para conexões sem fio.

A desvantagem desse tipo de topologia é a não redundância a falhas, já que se roteador central por algum motivo falhar ou ser danificado a rede inteira deixará de funcionar. Uma forma de remediar esse tipo de problema seria implementando uma segunda rede auxiliar que serviria de backup para a rede principal, ou se os custos não forem muito altos, a implantação de um sistema de *load-balance* para redundância de rede.

Figura 27 - Topologia de rede



Fonte: Autoria própria

5.3 Componentes Utilizados

- **ESP32 Wroom Devkit**

Dispositivo necessário para controlar a parte de hardware do projeto, foi escolhida uma opção mínima de tamanho para diminuir o tamanho do protótipo.

- Média de Preço: R\$ 50,00

- **Cabos (macho x fêmea) e (macho x macho)**

Necessário para a conexão de terminais da protoboard, e conexões nos pinos do ESP32

- Média de Preço: 20,00

- **Leds**

- Utilizados para sinalizar eventos nos dispositivos como estado de energia, tráfego de rede e sinal do sensor.

- Média de Preço: R\$ 5,00

- **Resistores**

- Ao todo foi utilizado 4 resistores de 110ohm, os resistores foram ligados em série para gerar um total de 220ohm e reduzir a tensão, como não são vendidos em quantidades menores é necessário comprar pacotes com cem resistores.

- Média de Preço: 20,00

- **Protoboard**

- Necessária para a montagem do protótipo, sem ela haveria uma desordem de fios e um possível dano a componentes frágeis que ao movimentar-se acabam quebrando.
- Média de Preço: R\$ 20,00
- **Sensor Ultrasônico HC-SR04**
 - Sensor responsável por realizar as aferições de distância do solo, apesar de pequeno consegue atingir grandes distancias, o que o torna ideal para pequenos projetos e prototipação.
 - Média de Preço: R\$ 20,00

6. RELATÓRIO

O projeto teve como principal objetivo o desenvolvimento de um dispositivo de monitoramento e visualização de dados gerados pelos sensores. Embora geralmente muitos dos projetos de embarcados sejam feitos utilizando a plataforma do Arduino, o dispositivo ESP32 apresentou vantagem e conseguiu manter a performance em ambientes e condições extremas como a alta taxa de requisições.

Durante o desenvolvimento do software do dispositivo, foram encontrados problemas e algumas incompatibilidades, a IDE do Arduino utilizada apresenta uma performance um pouco quanto lenta em ambientes Windows. Já em máquinas rodando sistemas UNIX esse problema não persiste, o mesmo arquivo compilado em ambiente Windows levou cerca de 1 minuto para ser compilado, enquanto o mesmo arquivo compilado em ambiente Linux (Debian) levou apenas 15 segundos, uma diferença gritante já que o código não possuía muitas linhas.

Outro ponto importante é o compilador, durante o desenvolvimento foram encontrados alguns problemas ao compilar arquivos utilizando os novos padrões C11, o compilador também teve problemas com variáveis que utilizam a *keyword* “extern”, bastante utilizada para referenciar variáveis localizadas em outros arquivos fontes.

A configuração do ESP32, não foi das mais fáceis, é necessário algumas alterações na IDE do Arduino para conseguir compilar e fazer o upload para a placa, como já apresentado na seção de desenvolvimento acima. Não existem bibliotecas pré-definidas que realmente funcionam, já que o mesmo chipset é utilizado por várias fabricantes e cada uma disponibiliza a sua. Por exemplo, o sensor HC-SR04 utilizado durante o processo de desenvolvimento não possuía suporte a IDE já que ela trabalha com apenas três pinos, já o sensor adquirido possui quatro, sendo necessário recorrer a bibliotecas e códigos de terceiros para conseguir executar funções e acessar o sensor.

Problemas de lentidão também foram encontrados durante o desenvolvimento do sistema, por ser um dispositivo simples, a maioria deles executam apenas uma tarefa por vez, ao tentar executar mais de uma ocorre um atraso de resposta que em muitos casos pode ocasionar um reboot do dispositivo, perdendo assim a conexão e em muitos casos os dados salvos na memória. A remediação encontrada para o problema foi a implementação e uso do sistema de multithread do dispositivo, tarefa um tanto quanto complicado pois o dispositivo roda em loop infinito, em casos em que

o sistema para de responder ou recebe um sinal de parada o dispositivo dispara o modo de *Kernel Panic*, quando isso ocorre a única solução é o reboot do dispositivo.

Figura 28 - Funções de Thread

```

44 void TaskHandleConnectionLed(void *pvParameters) {
45     (void)pvParameters;
46     for (;;) {
47         handleConnections();
48         blinkLed(SERVER_ONLINE);
49     }
50
51 }
52
53 void TaskDaemon(void *pvParameters) {
54     (void)pvParameters;
55     for (;;) {
56         sensorDaemon();
57         vTaskDelay(1000);
58     }
59 }
60

```

Fonte: Autoria própria

Outro problema encontrado foi a exibição de números de ponto flutuantes no dispositivo, que por padrão não existe uma função de arredondamento ou de casas decimais precisas, alguns deles, embora funcionem, apresentam comportamentos estranhos se o dispositivo ficar ligado por muito tempo, exibindo assim caracteres estranhos e números imprecisos. Como alternativa, resolveu-se enviar o número em formato normal no arquivo JSON e com a ajuda do Javascript do lado do cliente organizar as casas decimais e em muitos casos realizar o arredondamento dos números.

Figura 29 - Arquivo de retorno JSON

```

1 {
2     "device": "sensor12",
3     "status": "online",
4     "distance": 205.17099,
5     "safeDistance": 100,
6     "measureUnit": "cm",
7     "alert": 0
8 }

```

Fonte: Autoria própria

O sensor utilizado durante o projeto apresentou uma performance satisfatória, no entanto, existem algumas oscilações na rede que o fazem perder performance por um tempo. Apesar disso não há queda significativa na entrega de dados, ao operar em pulsos ele consegue se recuperar efetuando um novo pulso de leitura. Uma forma de contornar o problema é uma fonte de energia exclusiva para o sensor eliminando assim qualquer tipo de interferência ou oscilações causadas pela rede de alimentação da placa.

O único problema encontrado durante o desenvolvimento é que não pode ser sanado é a não presença do sensor. Em casos em que ele apresente defeitos ou pare de funcionar por problemas técnicos ou de conexão de seus terminais, é exibida uma mensagem de erro e o sistema de entrega de arquivos JSON para de funcionar, ao reconectar a alimentação do sensor, o dispositivo não o reconhece mais apresentando um comportamento estranho e emitindo erros. A única solução seria chamar o método novamente, no entanto, não é possível chamar uma função que já está ativa novamente, caso o faça o sistema para de responder. Sendo assim desligar e religar o ESP32 se torna a única opção viável.

O servidor de visualização de dados utilizando a linguagem Python não apresentou problemas que exigem uma atenção redobrada. No servidor se localizam apenas as rotinas de dados. Obtemos os dados do sensor através de uma requisição GET e essa informação é repassada a página web onde a lógica de programação e interpretação dos dados fica por conta do Javascript do lado do cliente.

As alterações feitas por meio do método POST são convertidos em objetos JSON e enviados ao servidor, garantido assim que as informações não sejam usadas de forma errônea. Porém usuários mais avançados podem interceptar a conexão entre o servidor e o dispositivo e realizar a leitura e modificação dos dados, uma vez que os dispositivos ESP32 não oferecem suporte decente a autenticação e criptografia de dados. Para mitigar o processo é importante não expor o dispositivo a internet ou redes externas, todas as requisições devem partir do servidor web.

Outro fator importante é o tempo em que o servidor fica ligado, ao realizar muitos acessos ao servidor ele pode apresentar lentidão ou ficar indisponível, o compilador Python ao encontrar um erro, pode não o tratar a tempo e encerrar a aplicação sem aviso, causando assim uma não disponibilidade. É interessante o uso

de uma ferramenta que detecta a ausência do script o reinicie novamente, isso pode ser feito em ambientes Unix utilizando *Shell Script*.

8. CONCLUSÃO

Através desse trabalho foi possível ter uma idéia sobre o que são dispositivos IoT (*Internet of Things*) e quais os benefícios de uso em casos específicos. Embora um pouco desconhecidos em nossos meios, tais dispositivos se tornam populares a cada dia na vida dos brasileiros e como no caso apresentado durante esse trabalho, se aplicado de forma correta pode evitar danos a vida e impactos ao meio ambiente.

Foi possível compreender o que são sistemas embarcados e quais os seus limites, suas aplicações e como programar tais dispositivos podem trazer aprendizado e ampliar as possibilidades de uso no desenvolvimento de software e hardware.

Graças a essas ferramentas e a *frameworks web*, foi possível criar um sistema de detecção de nível de água em tempo real, esse sistema pode auxiliar e alertar civis, empresas e governos sobre a elevação de água que seja ocasionada pelas chuvas, rompimento de comportas ou tanques de água.

Também se pode compreender e aprofundar os conhecimentos em desenvolvimento utilizando a linguagem de programação C, linguagem essa que embora seja um tanto quanto difícil de se aprender, se aplicada de forma correta apresenta uma performance e tempo de resposta muito alto, principalmente em sistemas embarcados onde o tempo de resposta é muito importante.

REFÊRENCIAS BIBLIOGRAFICAS

Biomania, O Histórico de Enchentes no Brasil: Causas e Tragédias. Biomania, 2019. Disponível em: <https://biomania.com.br/artigo/o-historico-de-enchentes-no-brasil-causas-e-tragedias>>. Acesso em: 02 de abril de 2022.

CNN. Os desastres naturais que impactaram o mundo recentemente. CNN, 2021. Disponível em: < <https://www.cnnbrasil.com.br/tecnologia/os-desastres-naturais-que-impactaram-o-mundo-recentemente/>>. Acesso em: 03 de abril de 2022.

Oracle. O que é IOT. Oracle, 2019. Disponível em: <<https://www.oracle.com/br/internet-of-things/what-is-iot/>>. Acesso em: 05 de abril de 2022.

Microsoft. O que é a IoT? Microsoft, 2019. Disponível em: < <https://azure.microsoft.com/pt-br/overview/internet-of-things-iot/what-is-the-internet-of-things/#overview>>. Acesso em: 05 de abril de 2022.

Cable, T. A Brief History of Network Technology. True Cable, 2022. Disponível em: <<https://www.truecable.com/blogs/cable-academy/a-brief-history-of-network-technology>>. Acesso em: 05 de abril de 2022.

Universidade Federal Fluminense. A Brief History of Computer Networking and the Internet, 1999. Disponível em: < http://www2.ic.uff.br/~michael/kr1999/1-introduction/1_09-history.htm>. Acesso em: 05 de abril de 2022.

Computer Science Engineering. Basics of Computer Network. Medium, 2019. Disponível em: <<https://medium.com/@computerscienceengineering/basics-of-computer-networking-6c7b961f4e14>>. Acesso em: 22 de abril de 2022.

Louisiana University. Networking Protocols, 2018. Disponível em: < <http://www2.southeastern.edu/Academics/Faculty/nadams/etec650/Protocols.html>>. Acesso em: 02 de maio de 2022.

Eshiett, Joseph. Computer Networking: What You Need To Know! Part 1. Medium, 2021. Disponível em: < <https://faun.pub/computer-networking-what-you-need-to-know-part-1-9e44d4b3dc40>> Acesso em: 10 de maio de 2022.

FICHAS DE ATIVIDADES PRÁTICAS SUPERVISIONADAS



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: BRUNO GONZALEZ MASSONE TURMA: CCSP39 RA: N582/J3
CURSO: CIÊNCIA DA COMPUTAÇÃO CAMPUS: ANCHIETA SEMESTRE: 1 TURNO: NOITE
CÓDIGO DA ATIVIDADE: 77B2 SEMESTRE: 2022/1 ANO GRADE: 2022

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
18/03	Organização do Trabalho	7h	BRUNO MASSONE		
20/03	Pesquisa sobre o tema	8h	BRUNO MASSONE		
21/03	Pesquisa sobre Arduino e	8h	BRUNO MASSONE		
25/03	Discussão	5h	BRUNO MASSONE		
28/03	Divisão de tarefas	5h	BRUNO MASSONE		
04/04	Introdução	5h	BRUNO MASSONE		
10/04	Referencial Teórico	5h	BRUNO MASSONE		
18/04	Desenvolvimento	5h	BRUNO MASSONE		
23/04	Desenvolvimento da visualização de dados	8h	BRUNO MASSONE		
30/04	Desenvolvimento ESP32	8h	BRUNO MASSONE		
05/05	Teste de Funcionamento	8h	BRUNO MASSONE		
14/05	Revisão do código	6h	BRUNO MASSONE		
23/05	Gravação de vídeo	2h	BRUNO MASSONE		
25/05	Ajustes	4h	BRUNO MASSONE		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.
Declaro que as informações acima são verdadeiras e que possuo os comprovantes a serem entregues ao coordenador do meu curso, no final do período de suspensão emergencial/Covid-19:

TOTAL DE HORAS ATRIBUÍDAS: _____
AVALIAÇÃO: _____ Aprovado ou Reprovado
NOTA: _____
DATA: ____/____/_____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Felipe de Oliveira Pereira Mauricio TURMA: CCSP39 RA: F2322A8
CURSO: CIÊNCIA DA COMPUTAÇÃO CAMPUS: ANCHIETA SEMESTRE: 1 TURNO: NOITE
CÓDIGO DA ATIVIDADE: 7782 SEMESTRE: 2022/1 ANO GRADE: 2022

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUIDAS (1)	ASSINATURA DO PROFESSOR
18/03	Organização do Trabalho	7h			
20/03	Pesquisa sobre o Tema	8h			
21/03	Pesquisa sobre Arduino e	8h			
25/03	Discussão	5h			
28/03	Divisão de tarefas	5h			
04/04	Introdução	5h			
10/04	Referencial Teórico	5h			
18/04	Desenvolvimento	5h			
23/04	Desenvolvimento da visualização de dados	8h			
30/04	Desenvolvimento ESP32	8h			
05/05	Teste de Funcionamento	8h			
14/05	Revisão do código	6h			
23/05	Gravação de vídeo	2h			
25/05	Ajustes	4h			

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

Declaro que as informações acima são verdadeiras e que possuo os comprovantes a serem entregues ao coordenador do meu curso, no final do período de suspensão emergencial/Covid-19:

TOTAL DE HORAS ATRIBUIDAS: _____

AValiação: _____ Aprovado ou Reprovado _____

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Felipe Nunes da Silva TURMA: CCS039 RA: N4935E0

CURSO: CIÊNCIA DA COMPUTAÇÃO CAMPUS: ANCHIEIA SEMESTRE: SEMESTRE: ANO GRADE: 2022 TURNO: NOITE

CÓDIGO DA ATIVIDADE: 7782 SEMESTRE: 2022/1

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
18/03	Organização do Trabalho	7h			
20/03	Pesquisa sobre o tema	8h			
21/03	Pesquisa sobre Arduino e	8h			
25/03	Discussão	5h			
28/03	Divisão de tarefas	5h			
04/04	Introdução	5h			
10/04	Referencial Teórico	5h			
18/04	Desenvolvimento	5h			
23/04	Desenvolvimento da visualização de dados	8h			
30/04	Desenvolvimento ESP32	8h			
05/05	Teste de Funcionamento	8h			
14/05	Revisão do código	6h			
23/05	Gravação de vídeo	2h			
25/05	Ajustes	4h			

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.
Declaro que as informações acima são verdadeiras e que possuo os comprovantes a serem entregues ao coordenador do meu curso, no final do período de suspensão emergencial/Covid-19.

TOTAL DE HORAS ATRIBUÍDAS: _____

AValiação: _____ Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO