

João Vitor Saade Simão

**Modelo de Monografia de
Trabalho de Conclusão de Curso com abnT_EX2**

Belo Horizonte

2025

João Vitor Saade Simão

Modelo de Monografia de Trabalho de Conclusão de Curso com abnT_EX2

Monografia apresentada durante o Seminário dos Trabalhos de Conclusão do Curso de Graduação em Engenharia Elétrica da UFMG, como parte dos requisitos necessários à obtenção do título de Engenheiro Eletricista.

Universidade Federal de Minas Gerais – UFMG

Escola de Engenharia

Curso de Graduação em Engenharia Elétrica

Orientador: Prof. Ricardo de Oliveira Duarte

Belo Horizonte

2025

*Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.*

Agradecimentos

Os agradecimentos principais são direcionados à Gerald Weber, Miguel Frasson, Leslie H. Watter, Bruno Parente Lima, Flávio de Vasconcellos Corrêa, Otavio Real Salvador, Renato Machnievscz¹ e todos aqueles que contribuíram para que a produção de trabalhos acadêmicos conforme as normas ABNT com L^AT_EX fosse possível.

Agradecimentos especiais são direcionados ao Centro de Pesquisa em Arquitetura da Informação² da Universidade de Brasília (CPAI), ao grupo de usuários *latex-br*³ e aos novos voluntários do grupo *abnTEX2*⁴ que contribuíram e que ainda contribuirão para a evolução do abnTEX2.

¹ Os nomes dos integrantes do primeiro projeto abnTEX foram extraídos de <<http://codigolivre.org.br/projects/abntex/>>

² <<http://www.cpai.unb.br/>>

³ <<http://groups.google.com/group/latex-br>>

⁴ <<http://groups.google.com/group/abntex2>> e <<http://abntex2.googlecode.com/>>

*“Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)*

Resumo

Segundo a ??, 3.1-3.2), o resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (...) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.

Palavras-chaves: latex. abntex. editoração de texto.

Abstract

This is the english abstract.

Key-words: latex, abnTeX, text editoration.

Lista de ilustrações

Figura 1 – Conexões básicas do ESP-12E	35
Figura 2 – Conexões do adaptador para ESP12-E	35
Figura 3 – Frente da placa de fenolite	36
Figura 4 – Verso da placa de fenolite	36
Figura 5 – Circuito interno do L298	37
Figura 6 – Placa original da Trato	49
Figura 7 – Placa de fenolite utilizada	49
Figura 8 – Placa original da Amanco	50
Figura 9 – Placa de fenolite utilizada	50

Lista de tabelas

Tabela 1 – Lista de casos de uso 33

Lista de abreviaturas e siglas

Fig. Area of the i^{th} component

CI Circuito integrado

IoT Internet of Things

List of symbols

Γ Greek letter Gamma

Λ Lambda

ζ Greek letter minuscule zeta

\in Pertains

Sumário

1	INTRODUÇÃO	23
1.1	Motivação e justificativa	23
1.2	Objetivos	23
1.3	Estado da arte	24
1.4	Relevância técnica	24
1.5	Ambiente de Desenvolvimento e Testes	25
1.6	Impacto e Contribuições do Projeto	25
1.7	Organização do trabalho	25
2	REVISÃO DE LITERATURA	27
2.1	Reaproveitamento	27
2.2	Ferramentas e Protocolos para Acesso Remoto a Dispositivos IoT	27
2.3	Alimentação	31
2.4	Controle do motor DC	32
2.5	Monitoramento do nível da bateria	32
3	METODOLOGIA	33
3.1	Levantamento de requisitos	33
3.2	Definição do método de programação do microcontrolador	33
3.3	Controle dos motores	36
3.4	Desenvolvimento do firmware	38
3.4.1	Conexão inicial do WiFi	38
3.4.2	Armazenamento das credenciais do WiFi	40
3.4.3	Criando um servidor	43
3.4.4	Programação Over-The-Air	44
3.4.5	Monitoramento da bateria	45
3.5	Desenvolvimento do aplicativo	46
3.5.1	Persistência das Configurações	46
3.6	Montagem	49
3.6.1	Irrigador Trato	49
3.6.2	Irrigador Amanco	49
3.7	Comentários	50
4	RESULTADOS E DISCUSSÃO	51
4.1	Vestibulum ante ipsum primis	51
4.2	Integer porta neque vitae massa	52

5	CONCLUSÕES	55
5.1	Pellentesque sit amet pede ac sem eleifend	55
5.2	Phasellus id magna	55
	 REFERÊNCIAS	 59
	 APÊNDICES	 63
	APÊNDICE A – QUISQUE LIBERO JUSTO	65
	APÊNDICE B – NULLAM ELEMENTUM URNA VEL IMPERDIET SODALES ELIT IPSUM PHARETRA	67
	 ANEXOS	 69
	ANEXO A – MORBI ULTRICES RUTRUM LOREM	71
	ANEXO B – CRAS NON URNA SED FEUGIAT CUM SOCIIS NATOQUE PENATIBUS	73
	ANEXO C – FUSCE FACILISIS LACINIA DUI	75

1 Introdução

1.1 Motivação e justificativa

Nos últimos anos, o lixo eletrônico tem se tornado um problema preocupante. Em 2019, 53,6 milhoes de toneladas métricas dele foram produzidas, sendo 7,3 kg per capita ([RENE et al., 2021](#)). Além do problema do imenso volume de resíduo produzido, o e-waste (nomenclatura em inglês) possui elementos extremamente prejudiciais à saúde humana e ao meio ambiente, como os metais pesados (??). Caso não sejam urgentemente tomadas medidas para mitigar esses efeitos, consequências graves para o meio ambiente e o ser humano ocorrerão ([RENE et al., 2021](#)).

Uma das soluções para reduzir essa quantidade imensa de lixo gerada e transformá-la em material útil é através da reciclagem. Segundo o site do Ministério do Meio Ambiente, "A reciclagem é um conjunto de técnicas de reaproveitamento de materiais descartados, reintroduzindo-os no ciclo produtivo"([RECICLAGEM](#),). Essa solução, porém, não é muito conveniente para o lixo eletrônico, já que, devido à sua complexidade, seu processo de reciclagem é extremamente desafiante e gera muitos resíduos tóxicos (??). Além disso, a reciclagem não trata o cerne do problema, que é alta geração de lixo eletrônico (??).

Fora a reciclagem, existe um outro processo chamado upcycling, que consiste em dar uma nova função a um objeto considerado descartável ([ABOUT..., 2017](#)). Embora tenha sido aplicado inicialmente na indústria da moda para o reaproveitamento de tecidos antigos, esse conceito é aplicado em diversas áreas, inclusive na eletrônica(??). A vantagem desse método é que ele permite a reutilização do material sem que sejam necessários processos caros ou tóxicos, além de postergar o descarte desses materiais.

Neste trabalho, propõe-se aplicar o conceito de upcycling a dois sistemas de irrigação defeituosos, com o objetivo de evitar seu descarte e transformá-los em um novo dispositivo, mais funcional e com maior valor agregado. Essa abordagem visa não apenas reduzir o impacto ambiental, mas também promover uma cultura de reaproveitamento na área da eletrônica.

1.2 Objetivos

Este trabalho tem como objetivo o reaproveitamento de dois sistemas de irrigação com defeito: um da marca Amanco (sem identificação de modelo) e outro da marca Trato, modelo TD16 ([TRATO...,](#)). O primeiro apresentou falhas no visor após exposição prolongada às intempéries (sol, chuva, etc.); o segundo deixou de funcionar corretamente

após o esgotamento da bateria, não retomando sua operação mesmo após a substituição da mesma. Esses problemas evidenciam falhas de projeto em ambos os dispositivos, que resultaram em falhas prematuras e prejuízos para o usuário.

Com o intuito de corrigir essas deficiências, serão incorporadas novas funcionalidades aos dispositivos, visando aumentar sua vida útil e proporcionar uma experiência de uso mais confiável e eficiente.

1.3 Estado da arte

Com o objetivo de buscar referências para este trabalho, foi realizada uma pesquisa na internet sobre o reaproveitamento de sistemas de irrigação. No entanto, não foram encontrados projetos que abordassem especificamente esse tema. A maioria das soluções encontradas tratava da construção de sistemas de irrigação automáticos inteiramente novos, sem foco no reaproveitamento de dispositivos já existentes ([SANTOS, 2024](#); [SADDAM, 2016](#); [SriTu Hobby, 2023](#)).

Diante dessa lacuna no estado da arte, este trabalho propõe o desenvolvimento de um projeto voltado para o reaproveitamento dos dois temporizadores de irrigação mencionados anteriormente. O procedimento completo será disponibilizado na plataforma GitHub, permitindo o acesso livre e facilitado por outros interessados. Serão compartilhadas instruções detalhadas sobre a montagem do hardware, incluindo sua fixação nas carcaças originais dos dispositivos, bem como os códigos-fonte utilizados tanto no microcontrolador quanto no aplicativo de controle.

1.4 Relevância técnica

A utilização de displays LCD nos dispositivos de irrigação apresenta limitações técnicas significativas. Esses componentes não foram projetados para exposição direta e prolongada à luz solar intensa, o que compromete sua durabilidade ([ELSYSCOM, 2018](#); [SONY, 2025](#)). Além disso, a visibilidade em ambientes externos é prejudicada, pois o reflexo em suas superfícies dificulta a leitura das informações, tornando-os inadequados para uso sob luz solar direta - especialmente no ambiente em que esses sistemas são implantados, como hortas, jardins e áreas externas expostas às intempéries.

Outro ponto relevante é a limitação imposta pela necessidade de interação física com os dispositivos. A presença dos displays diretamente no equipamento exige que o usuário esteja próximo ao sistema sempre que for necessário visualizar seu estado ou realizar configurações. Isso representa um obstáculo em cenários típicos de uso, como hortas ou jardins de difícil acesso. A adoção de interfaces sem fio, como a comunicação via Wi-Fi, surge como uma alternativa vantajosa, permitindo o monitoramento e controle

remoto do sistema. Tal abordagem não apenas melhora a experiência do usuário, mas também aumenta a praticidade e a eficiência operacional do dispositivo em campo.

1.5 Ambiente de Desenvolvimento e Testes

O processo de montagem e teste dos protótipos, assim como a construção do produto final, será realizado tanto nos laboratórios da UFMG (FabLab e PETEE) ? principalmente quando é necessário o uso de equipamentos como fontes de tensão, geradores de sinais e osciloscópios ? quanto em ambiente doméstico, quando os testes podem ser feitos sem a necessidade desses recursos.

Para a realização dos testes iniciais, serão componentes emprestados do grupo Programa de Educação Tutorial da Engenharia Elétrica (PETEE), os quais serão devidamente devolvidos ao término dessa etapa. Posteriormente, os componentes definitivos serão adquiridos em lojas de eletrônica. O teste final de funcionamento do dispositivo será conduzido em ambiente doméstico, refletindo a aplicação prática para a qual o sistema foi projetado.

1.6 Impacto e Contribuições do Projeto

Como resultado deste projeto, espera-se que usuários dos sistemas de irrigação abordados - e até mesmo de dispositivos similares - possam reaproveitar o desenvolvimento proposto em seus próprios equipamentos. Além disso, acredita-se que este trabalho possa inspirar outros projetistas a desenvolverem soluções semelhantes para diferentes tipos de dispositivos eletrônicos.

Espera-se, com isso, fomentar uma cultura de reutilização tecnológica (upcycling), contribuindo para a redução do lixo eletrônico e para o aumento da vida útil de equipamentos. Adicionalmente, há a expectativa de que este projeto desperte maior interesse da população pela área de eletrônica, seja como hobby ou como profissão.

1.7 Organização do trabalho

O texto do trabalho está organizado da seguinte forma:

- O capítulo 2 apresenta uma revisão de projetos similares, destacando suas semelhanças e diferenças em relação ao objetivo deste trabalho.
- O capítulo 3 descreve as decisões de projeto, a definição dos requisitos e os processos de montagem e programação do sistema.

- O capítulo 4 traz os resultados obtidos nos testes e no produto final, discutindo em que medida os objetivos propostos foram alcançados.
- O capítulo 5 aponta os possíveis desdobramentos e sugestões para trabalhos futuros.

2 Revisão de Literatura

2.1 Reaproveitamento

É possível encontrar na internet alguns projetos de robótica e automação que envolvem a reutilização de materiais descartados ([HOFFMANN, ; OLIVEIRA, 2019](#)). Nesses projetos, materiais domésticos descartados eram utilizados como carcaça dos equipamentos. Embora seja uma excelente ideia para reduzir o lixo nas casas e tornar os projetos eletrônicos mais acessíveis, essa solução é diferente daquela proposta nesse trabalho, que consiste no reaproveitamento de materiais dos próprios equipamentos eletrônicos defeituosos.

2.2 Ferramentas e Protocolos para Acesso Remoto a Dispositivos IoT

Para comunicação do microcontrolador com o usuário, foi encontrado um projeto de irrigação automática que utilizava o SMS para transmitir informações do estado do sistema ([SADDAM, 2016](#)). Nesse projeto, foi utilizado o Arduino Uno para controlar um relé, que por sua vez acionava uma bomba de água. Por não possuir interface de usuário para controle e programação da irrigação, a comunicação por SMS não será adotada nesse TCC. Outro fator a se mencionar é que essa forma de comunicação não é comumente utilizada hoje em dia, sendo mais difícil o usuário se adaptar.

Um meio muito utilizado nos projetos de IoT para realizar o controle e o monitoramento deles é através de plataformas Cloud. Entre elas, encontramos projetos que usaram Blynk IoT Cloud ([IoT Projects Ideas, 2022](#)), Arduino IoT Cloud ([Education is Life \(joed goh\), 2022a](#)) e Azure IoT ([IoT Frontier, 2023](#)).

Em relação aos trabalhos utilizando Blynk IoT Cloud, o desenvolvimento de projetos nessa plataforma mostrou-se relativamente simples, sendo estes os passos para realizar o projeto:

1. Registrar-se no site da Blynk IoT;
2. Acessar dispositivos → Novo modelo;
3. Adicionar Datastreams;
4. Editar a interface em Painel de Controle da Web;

5. Acessar Dispositivos → Novo dispositivo → A partir do modelo;
6. Copiar o token fornecido e colar no código a ser usado;
7. Editar o código e fazer upload no microcontrolador.

A plataforma em questão também oferece em seu site vários modelos de projeto que podem ser copiados e aproveitados. Além disso, o tempo de resposta é baixo e é possível acessar o dispositivo tanto via Web quanto via aplicativo de smartphone.

Contudo, existem algumas limitações importantes nessa plataforma:

- A versão gratuita só permite uma tela.
- A versão gratuita não possui Widgets para programação de tempo.

Considerando que neste TCC há a necessidade de uma tela para editar a programação e outra para ativar a programação desejada, a utilização dessa plataforma foi descartada.

Em relação à plataforma Arduino IoT Cloud, foram encontradas algumas diferenças. O processo de registro e desenvolvimento de projetos nessa plataforma consiste nos seguintes passos:

1. Instalar o software Arduino Create Agent;
2. Clicar no ícone do Arduino na barra de ícones do Windows;
3. Fazer Log In no site do Arduino.cc;
4. Criar um "Thing";
5. Adicionar as variáveis de entrada e saída a serem utilizadas;
6. Clicar em associar dispositivo;
7. Copiar o Device ID e Secret Key;
8. Colocar os dados de Wi-Fi;
9. Mudar o código conforme desejado (na seção Sketch);
10. Editar o dashboard (como o aplicativo vai aparecer no celular);
11. Fazer o upload do código em "Sketch".

Embora a solução na plataforma Arduino IoT Cloud envolva mais passos que na plataforma Blynk IoT Cloud, ela oferece uma facilidade maior por permitir que o usuário edite o código e passe para o dispositivo na própria página Web. Em razão disso, essa plataforma é ideal para que usuários inexperientes na área desenvolvam seu primeiro projeto IoT. Contudo, apesar da série de benefícios da plataforma Arduino IoT Cloud, ela não oferece uma interface ideal para o projeto de irrigador automático desejado nesse trabalho, que necessita de programar e editar timers para o acionamento programado do dispositivo.

Já o Azure IoT ([IoT Frontier, 2023](#)) representa uma solução mais robusta, fornecida pela Microsoft. Trata-se de um conjunto de serviços em nuvem voltado para conectar, monitorar e controlar dispositivos embarcados. Oferece alto nível de segurança, com suporte a múltiplos protocolos de autenticação e integração com outras ferramentas da Microsoft. Apesar de suas vantagens, essa solução foi descartada por apresentar limitações significativas para o presente projeto: exige conexão constante com a internet (não operando em rede local), possui maior complexidade de configuração e exige assinatura paga para acesso completo aos recursos, o que contraria os objetivos de simplicidade e baixo custo do irrigador automático.

Existem também projetos que utilizam o protocolo MQTT ([SANTOS, 2020](#)). Nesses projetos, é comum o uso da ferramenta Node-RED para a visualização dos dados. O MQTT (Message Queuing Telemetry Transport) é um protocolo leve de comunicação assíncrona, baseado no modelo publicador/assinante, amplamente utilizado em aplicações de Internet das Coisas (IoT) por sua eficiência na transmissão de pequenas quantidades de dados entre dispositivos. No entanto, esse tipo de abordagem não foi considerada adequada para o projeto do irrigador automático, pois requer a presença de um broker ? um servidor intermediário responsável por gerenciar a troca de mensagens entre os dispositivos. Como o foco deste projeto é desenvolver uma solução de baixo custo e com poucos sensores de leitura, a utilização do MQTT foi descartada.

Outro protocolo encontrado em projetos de IoT é o Matter ([ESP32..., 2024](#); [GET..., 2023](#)). O Matter é altamente flexível e compatível com diversos padrões de comunicação sem fio, como Wi-Fi, Thread e Ethernet. Entre suas vantagens, destaca-se a facilidade de integração entre diferentes dispositivos e a criação de uma experiência de usuário unificada. No entanto, para seu funcionamento completo, é necessário o uso de um hub central ? como o Amazon Alexa, Google Nest ou Apple HomePod ? que atua como controlador e intermediador entre os dispositivos. Considerando que o projeto do irrigador automático visa uma solução de baixo custo e independente de infraestrutura adicional, a adoção do protocolo Matter foi considerada inviável para este trabalho.

Outra plataforma bastante utilizada em projetos de IoT é o Firebase ([Education is Life \(joed goh\), 2022b](#)), serviço de backend as a service (BaaS) oferecido pelo Goo-

gle. O Firebase oferece uma série de funcionalidades que facilitam a comunicação entre dispositivos e aplicativos, como banco de dados em tempo real (Firebase Realtime Database), autenticação, notificações e hospedagem. Ele é especialmente útil em projetos que requerem sincronização instantânea de dados entre múltiplos clientes.

Uma limitação importante do Firebase é que ele exige conexão com a internet, pois todos os dados são armazenados e acessados em servidores remotos do Google. Dessa forma, não é possível utilizá-lo em uma rede local sem internet. Isso representa uma desvantagem significativa para projetos que precisam operar de forma autônoma ou em ambientes sem conectividade, como pode ser o caso de sistemas de irrigação em áreas remotas. Por esse motivo, a plataforma Firebase foi considerada inadequada para o presente trabalho, que requer uma solução independente de conexão constante com a nuvem.

Uma ferramenta amplamente utilizada no contexto de IoT é a ThingSpeak ([Robu.in, 2024](#); [IoT Frontier, 2024](#)). Essa plataforma de código aberto permite o armazenamento, visualização e análise de dados enviados por dispositivos conectados. Um de seus principais diferenciais é a integração nativa com o MATLAB, o que facilita a aplicação de algoritmos avançados de processamento e análise de grandes volumes de dados. A ThingSpeak também oferece uma variedade de recursos para visualização, como gráficos em tempo real, indicadores de temperatura e interfaces personalizáveis para monitoramento remoto. No entanto, no contexto deste projeto, que lida com um volume reduzido de dados ? restrito à leitura do nível da bateria ? e não requer análise estatística ou processamento avançado, a utilização da plataforma foi considerada desnecessária. Além disso, seu uso implicaria em uma complexidade adicional na implementação, contrariando o objetivo do projeto, que é fornecer uma solução simples, acessível e de fácil uso, mesmo por usuários sem experiência técnica prévia.

Outra ferramenta relevante para projetos IoT, especialmente no desenvolvimento da interface com o usuário, é o Android Studio ([CREATE...,](#)). Trata-se do ambiente oficial de desenvolvimento de aplicativos Android, mantido pelo Google, que permite a criação de aplicações móveis completas e altamente personalizáveis. A principal vantagem dessa abordagem está em sua flexibilidade, pois o desenvolvedor tem controle total sobre a interface gráfica, a lógica de programação e a forma como os dados são enviados e recebidos ? seja por Wi-Fi, Bluetooth, ou outras interfaces de comunicação.

Embora o uso do Android Studio não seja trivial, especialmente para iniciantes, ele se torna uma excelente escolha quando o projeto demanda uma interface específica, que não pode ser facilmente implementada com plataformas prontas. No caso do irrigador automático, por exemplo, é necessário que o aplicativo permita configurar horários, dias da semana e duração da irrigação de forma intuitiva ? funcionalidades que não são comumente encontradas em soluções prontas, como Blynk, Arduino IoT Cloud ou Firebase.

Além disso, ao desenvolver um aplicativo próprio, o sistema pode ser ajustado

para funcionar com ou sem conexão com a internet, operando diretamente em rede local se necessário. Isso se alinha aos objetivos do projeto, que busca oferecer uma solução independente, de baixo custo e funcional em ambientes com conectividade limitada. Por essas razões, optou-se pelo uso do Android Studio como ferramenta para criar a interface de controle do irrigador.

Outro protocolo amplamente utilizado em aplicações IoT é o HTTP (Hypertext Transfer Protocol) ([CONSTRUINDO..., 2025](#); [SANTOS, 2024](#); [ESP32..., 2025](#)). Trata-se de um protocolo de comunicação baseado em requisições e respostas, onde um cliente (como um aplicativo ou navegador) envia uma requisição para um servidor (como um microcontrolador conectado à rede), que por sua vez retorna uma resposta. O HTTP é simples de implementar e é compatível com praticamente todas as plataformas e linguagens de programação. Além disso, é possível utilizá-lo em redes locais sem a necessidade de conexão com a internet, o que representa uma vantagem importante para sistemas que precisam funcionar offline, como o irrigador automático.

No contexto deste projeto, o uso do protocolo HTTP se mostra bastante vantajoso, pois permite que um aplicativo desenvolvido no Android Studio envie comandos diretamente ao microcontrolador (como um ESP8266) via rede Wi-Fi local. Isso possibilita, por exemplo, o envio de configurações de tempo de irrigação, bem como comandos de ativação ou desativação do sistema. O protocolo também permite que o microcontrolador atue como um servidor web, recebendo requisições HTTP e respondendo com confirmações ou dados, como o nível da bateria ou o status da irrigação.

Considerando esses fatores, o protocolo HTTP foi considerado adequado e viável para o projeto, atendendo aos requisitos de simplicidade, flexibilidade e operação em rede local.

2.3 Alimentação

Foi realizada uma pesquisa sobre como é realizada a alimentação dos microcontroladores nos projetos IoT. Na maioria dos projetos encontrados ([SANTOS, 2024](#); [CONSTRUINDO..., 2025](#)), não é informada a forma de alimentação definitiva do projeto, em razão de serem projetos de prototipagem ou cuja função é unicamente de aprendizado. Nesses projetos, os dispositivos são alimentados pela interface USB conectada ao computador, que também serve para a programação dos microcontroladores.

Contudo, foi encontrado um projeto utilizando baterias de lítio de 3,7V para alimentar o microcontrolador ([IoT Projects Ideas, 2022](#)). Nesse projeto, também é utilizado o módulo TP4056 para recarregamento da bateria 18650, utilizada no projeto. Embora esta seja uma solução bastante útil, por permitir a recarga do dispositivo, ela é incompatível com o projeto do irrigador, pois este deve ser projetado para ser alimentado por

baterias comuns (alcalinas) para tornar o dispositivo mais acessível e prático.

Existem também projetos que utilizam fontes de tensão de 12V ([SADDAM, 2016](#)). Nesse projeto, foi utilizado um regulador de tensão LM317 para reduzir a tensão para um nível mais baixo para o funcionamento adequado dos dispositivos. Outro projeto de irrigação também utilizou esse tipo de fonte foi feito pela Robocore ([ROBOCORE](#),). Porém, por se tratar da placa Arduino Uno, que já possui esse tipo de entrada a alimentação com a fonte de 12V é comum. Esse tipo de alimentação não é próprio para o projeto do irrigador automático, considerando que ele deve ser capaz de funcionar em áreas onde não há acesso a tomadas, além de ser necessário utilizar um microcontrolador menor para caber na carcaça reaproveitada do irrigador.

2.4 Controle do motor DC

A maioria dos projetos analisados emprega o uso de ponte H como principal método para o controle de motores DC, dada sua eficiência na reversão do sentido de rotação e na modulação da velocidade por PWM ([SANTOS, 2024](#)). No entanto, também foram encontradas abordagens alternativas, que utilizam circuitos discretos (utilizando transistores bipolares de potência) ([How...,](#)) ou relés (para motores maiores) ([SADDAM, 2016](#)).

Considerando que o motor usado é de baixa tensão e que o tamanho é um requisito primordial no projeto, foi preferido utilizar uma ponte H para realizar o controle do motor DC do irrigador.

2.5 Monitoramento do nível da bateria

Para o monitoramento do nível de bateria, o método mais encontrado é através do divisor de tensão ([IoT Projects Ideas, 2022](#); [EDISON SCIENCE CORNER, 2023](#)). Nesses sistemas, são selecionados resistores com alta resistência a fim de drenar a menor quantidade de carga possível. A utilização do divisor de tensão se deve ao fato de que muitos conversores analógicos digitais aceitam entradas de tensão com valores máximos que são menores que aqueles usados para alimentação. No microcontrolador utilizado no projeto do irrigador automático, o ESP8266, a tensão máxima no ADC é de 1V.

Por se tratar de uma solução simples e que não utiliza muita corrente, esta foi a opção escolhida para o projeto.

3 Metodologia

3.1 Levantamento de requisitos

Para obter os requisitos aos quais o sistema deve atender para cumprir com seu objetivo, foram utilizadas técnicas de Engenharia de Requisitos. Para o levantamento inicial dos requisitos foi realizada uma entrevista com o professor orientador, cujo papel é de cliente e usuário do sistema a ser desenvolvido. A tabela 1 apresenta a lista de casos de uso que foram definidos:

Tabela 1 – Lista de casos de uso

Nº	Caso de uso	Descrição
1	Adicionar configuração.	Adiciona o horário, duração e dias da semana de irrigação.
2	Excluir configuração.	Deleta as configurações do banco de dados e no microcontrolador.
3	Verificação de status.	Envia mensagem para o microcontrolador e espera resposta para indicar correto funcionamento.
4	Editar configuração.	Escolhe uma configuração já criada e altera seus dados.
5	Verificar configurações.	Pergunta ao microcontrolador quais configurações estão salvas nele.
6	Verificar bateria.	Pergunta ao sistema se a bateria está em nível alto, médio ou baixo.

Fonte: Produzido pelo autor.

Nota: Esta tabela segue o modelo fornecido pelo livro "Engenharia de Software: fundamentos, métodos e padrões".(FILHO, 2009)

3.2 Definição do método de programação do microcontrolador

O microcontrolador, devido a seu tamanho reduzido, não possui entrada USB, sendo necessário recorrer a diferentes meios para a programação do mesmo. Entre as possíveis alternativas, foram levadas em consideração quatro ([witnessmenow](#), ; [Indrek](#), 2020; [PROGRAMMING...](#), 2017):

- a) placa de programação dedicada;
- b) outro microcontrolador (NodeMCU);
- c) adaptador USB para serial;
- d) sem fio (Wi-Fi).

A utilização de uma placa dedicada é uma excelente opção para um projeto de fácil execução, mesmo para usuários inexperientes em eletrônica. Contudo, levando em conta que acrescentar esse item ao projeto aumentaria os custos do mesmo em torno de R\$ 80,00, essa opção foi desconsiderada.

A opção de utilizar outro microcontrolador consiste em desabilitar o microcontrolador em uma placa como o NodeMCU e aproveitar os sinais vindos do USB presente nele para programar o Módulo Wi-Fi. Essa opção traz um menor custo que a anterior, porém aumenta a dificuldade de realização.

Existe também a opção de utilizar somente um adaptador que transforma a interface UART do Módulo na interface USB, própria para a programação. Para isso, são necessárias duas pilhas de 1,5V e alguns jumpers, necessários para a comunicação do adaptador com o módulo e a configuração deste. Nessa terceira opção, a dificuldade de realizar a programação não é significativamente maior que a anterior, e introduz menos custos, sendo ideal para esse projeto.

Portanto, considerando que o objetivo do projeto é montar um produto de baixo custo e de fácil construção, foi escolhido utilizar o adaptador USB para serial.

Para realizar a programação utilizando um adaptador USB serial, são necessárias algumas conexões:

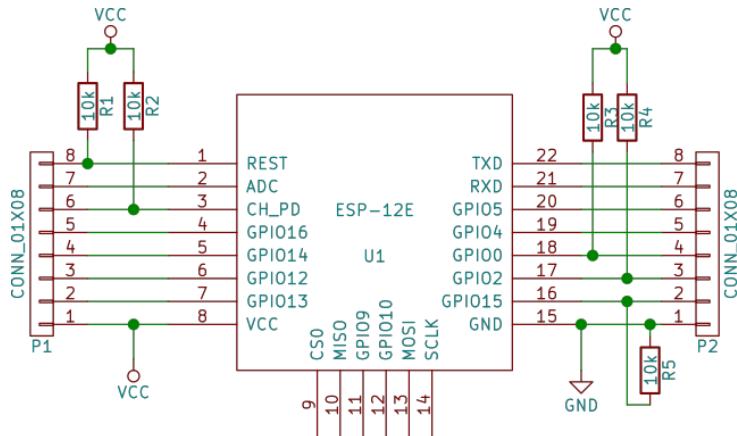
1. O RX do microcontrolador deve ser conectado ao TX do adaptador;
2. O TX do microcontrolador deve ser conectado ao RX do adaptador;
3. O GND do microcontrolador deve ser conectado ao GND do adaptador;
4. O RST do microcontrolador deve ser conectado ao GND através de um botão;
5. O GPIO0 do microcontrolador deve ser conectado ao GND através de um botão;
6. O GPIO15 do microcontrolador deve ser conectado ao GND através de um resistor de $10k\Omega$;
7. O CH_PD(EN) do microcontrolador deve ser conectado ao VDD através de um resistor de $10k\Omega$.

A imagem 1 mostra essas conexões.

Em relação às últimas duas conexões, a do GPIO15 e a do CH_PD(EN), elas já estão presentes na placa adaptadora usada no irrigador da Amanco. A imagem 2 mostra essas conexões.

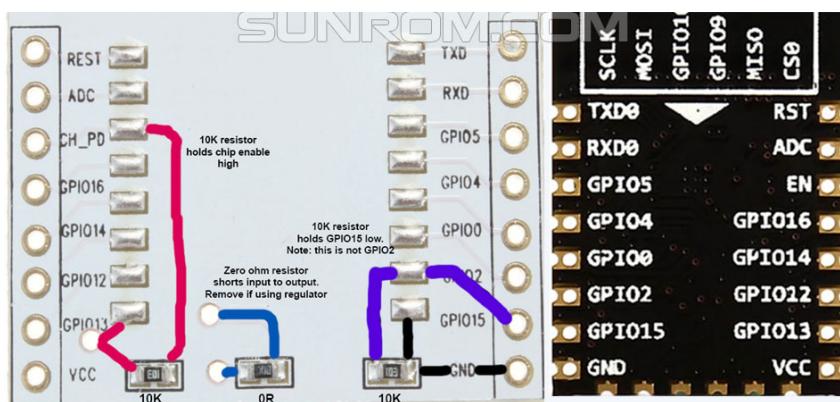
Para realizar a programação do módulo, o botão conectando o GPIO0 ao GND deve ser segurado. Ao fim da programação, o botão deve ser solto e o botão que liga o

Figura 1 – Conexões básicas do ESP-12E.



Fonte: Small Bits ([MARSHALL, 2017](#)).

Figura 2 – Conexões do adaptador para ESP12-E



Fonte: Sunrom ([BREAKOUT...,](#)).

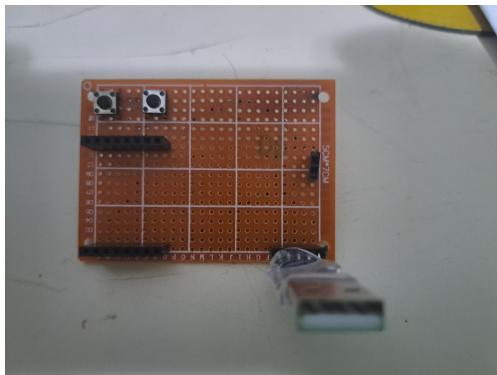
RST ao GND deve ser pressionado e solto. Assim, foi realizada a primeira programação do módulo. As programações subsequentes foram realizadas pelo método Over the Air (OTA).

Para facilitar a programação do módulo, todos os componentes necessários para a programação do circuito foram montados em uma placa de fenolite perfurada. Para realizar a programação do módulo, o microcontrolador deveria ser encaixado no local designado, duas baterias de 1,5V seriam conectadas através de jumpers, e a interface USB deveria ser conectada no computador desejado para a programação.

As figuras 3 e 4 mostram a visão superior e inferior da placa, respectivamente.

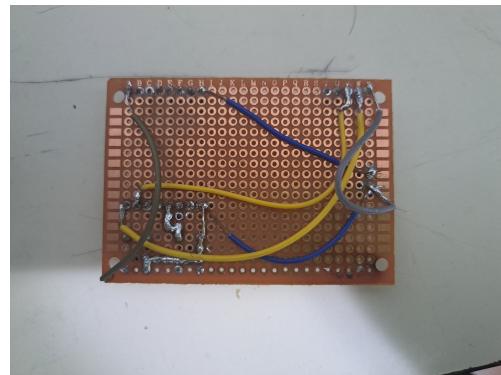
Em relação à programação via WiFi, ela é bastante interessante de ser feita por permitir que o firmware do microcontrolador seja atualizado mesmo quando o dispositivo já esteja soldado na placa definitiva. Para que isso seja possível, é necessário utilizar a funcionalidade Over-The-Air (OTA) do ESP. Os códigos para isso são descritos na seção

Figura 3 – Frente da placa de feno-lite



Fonte: Autoria própria

Figura 4 – Verso da placa de feno-lite



Fonte: Autoria própria

3.4.4.

3.3 Controle dos motores

Para fazer o controle dos motores, é necessário verificar qual a corrente consumida pelo motor e qual o máximo de corrente que o microcontrolador consegue fornecer. Embora não tenha sido achada indicação do modelo de motor utilizado nos irrigadores, motores semelhantes alimentados com 3V consomem por volta de 150mA ([MINI...^a](#)). Em relação ao microcontrolador, no site da Expressif, foi informado que a corrente máxima de saída de cada pino é de 12mA ([GPIO...^b](#)). Consequentemente, não é possível acionar o motor diretamente com a corrente do microcontrolador, sendo necessário utilizar um amplificador de corrente.

Uma opção para realizar essa tarefa é utilizando um transistor Darlington TIP122 ([How...^c](#)). Nessa solução, o microcontrolador aciona o BJT com uma corrente de base, que o aciona e permite a passagem de corrente por ele.

Para verificar se esse transistor é capaz de fornecer a corrente necessária, é preciso verificar qual o ganho de corrente que ele é capaz de produzir. Segundo o datasheet do TIP122 ([TIP122...^d, 2025](#)), o ganho máximo que ele é capaz de fornecer é de 1000mA/mA, bem superior ao necessário para o funcionamento do motor, que seria de 20 a 40.

Apesar de ser uma solução barata (por volta de R\$ 3,00) e suportar uma alta corrente (8A ([TIP122...^e, 2025](#))), ela é mais difícil de ser feita, já que necessitaria de soldas e isso tornaria a tarefa mais difícil para alguém inexperiente na área.

Outra alternativa para solucionar esse problema é a utilização de uma ponte H, um circuito eletrônico que permite controlar a direção da corrente fornecida a uma carga, como um motor de corrente contínua. Esse dispositivo possui dois terminais de controle de entrada, que determinam o sentido de rotação do motor, e dois terminais de saída,

responsáveis por fornecer a tensão e a corrente necessárias ao seu funcionamento.

A figura 5 mostra o circuito interno do CI L298, utilizado na ponte H.

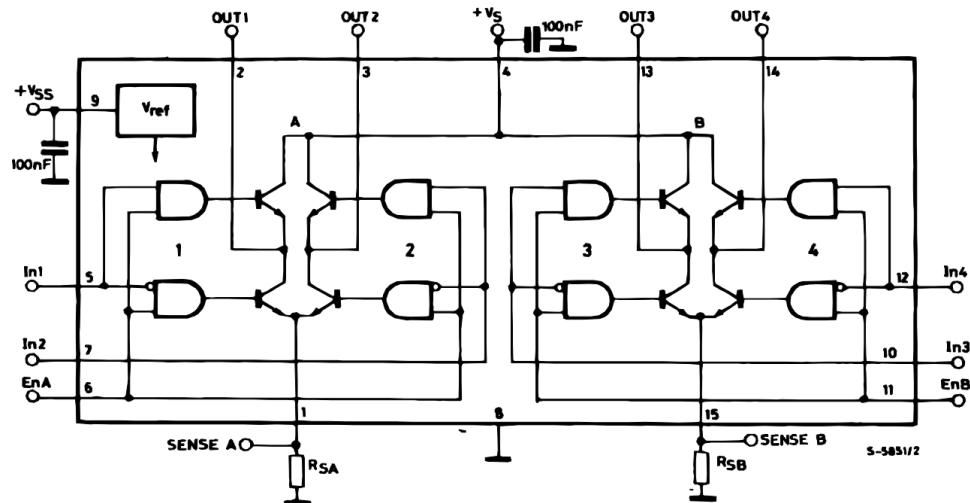


Figura 5 – Circuito interno do L298

O funcionamento desse circuito consiste em conectar os pinos de saída ao terra ou à alimentação positiva. Quando a entrada tem valor alto, o transistor BJT superior conecta a alimentação positiva à saída, enquanto o transistor inferior permanece desligado. Por outro lado, quando o nível da entrada é baixo, o transistor inferior é acionado, conectando a saída ao terra.

Ao inserir pares de entrada alternados ($In1=1$ e $In2=0$, por exemplo), uma das saídas é ligada ao terra e outra à alimentação positiva, criando uma diferença de potencial no motor e fazendo uma corrente circular nele. É importante ressaltar que ao utilizar a ponte H, a corrente que sai do microcontrolador somente é utilizada para acionar o transistor, como corrente de base deste. A corrente utilizada para o funcionamento do motor, por sua vez, tem origem na fonte de alimentação, passando pelos dois transistores e pelo motor.

Essa solução é a mais apropriada para o projeto, já que não acrescenta muitos custos (por volta de R\$5,00) e é mais fácil de ser montada e controlada. Outra vantagem importante é que a ponte H pode alterar facilmente o sentido da corrente no motor, algo essencial nesse projeto, considerando que o motor deve obrigatoriamente girar nos dois sentidos.

A fim de confirmar se a ponte H consegue realizar esse papel, é necessário investigar quando de corrente ela é capaz de fornecer, e quanto de corrente de entrada ela exige.

Segundo informação do datasheet do CI L298, a corrente máxima de saída para a operação DC é de 2A (L298..., 2025). Em relação à placa usada, a mini ponte H, o site no qual ela foi comprada informa que a corrente máxima por canal é de 1,5 A (MINI...,

b). Considerando que a corrente exigida pelo motor DC é de 150mA - 200mA, a ponte H utilizada é capaz de fornecer com folga a corrente de saída.

Em relação à corrente de entrada, o datasheet informa que para uma entrada de nível alto, a corrente consumida é de no máximo $100\mu A$, enquanto para uma entrada de nível baixo o consumo é de $-10\mu A$, estando, portanto, dentro das restrições de corrente do microcontrolador.

Outro modelo de ponte H possível para ser utilizado nesse projeto é o DRV8833. Seu funcionamento é semelhante ao explicado para o modelo anterior. Diferente do driver L298, o DRV8833 possui transistores MOSFET como entrada, além de dois pinos extras: SLEEP e FAULT. Este primeiro pino é uma entrada e possui a funcionalidade de desativar as saídas caso esteja em valor baixo; o segundo pino, por sua vez, é uma saída, e fica em valor baixo quando há uma sobre-corrente ou sobre-temperatura. Essa segunda funcionalidade da ponte H é bastante interessante para indicar ao usuário quando há algum defeito no motor.

As dimensões dos módulos de ponte H citados anteriormente são:

- **L298:** 25mm x 21mm x 7mm
- **DRV8833:** 18,5mm x 16,3mm 3,2mm

Considerando que o tamanho dos componentes é uma limitação importante do projeto, e levando em consideração a funcionalidade de indicação de sobrecorrente e sobretemperatura, a ponte H DRV8833 foi escolhida para o projeto.

3.4 Desenvolvimento do firmware

3.4.1 Conexão inicial do WiFi

O primeiro passo para o desenvolvimento do firmware do microcontrolador consistiu em desenvolver uma forma de conectar o dispositivo a qualquer rede, sem a necessidade de fazer upload de um novo código. Para que isso fosse realizado, era necessário recorrer a uma das três técnicas: SmartConfig, WPS, ou Local AP. Para o microcontrolador ESP32, também existe a técnica do BluFi, na qual o bluetooth é usado para facilitar a primeira conexão com o WiFi. Como o microcontrolador usado nesse projeto, o ESP8266, não possui essa funcionalidade, essa opção foi descartada.

O Wi-Fi Protected Setup (WPS) é um padrão utilizado para a conexão de dispositivos sem a necessidade de senha. Ele possui a grande vantagem de ser simples, necessitando apenas de apertar o botão do roteador. Esse método possui algumas questões em relação à segurança e ao acesso indevido de usuários indesejados. Contudo, como esse trabalho não

possui foco em desenvolver um sistema protegido de acessos indevidos, essa característica não o impediria de ser utilizado. Contudo, a maior desvantagem desse método é que para se conectar a rede deve se estar próximo do roteador e ter acesso físico a ele, o que pode não ser possível em determinados casos. Além disso, essa tecnologia não se encontra em todos os roteadores comercializados, limitando bastante sua aplicação. Devido a essas razões, optou-se por não utilizar essa tecnologia.

Na técnica de Local AP, o microcontrolador torna-se um ponto de acesso (Access point) ao qual um dispositivo deve se conectar e, através de uma interface web, inserir o SSID e a senha de uma das redes disponíveis (FREITAS, 2021). Tendo em vista que o modo de Local AP exige um maior gasto de energia e que o projeto desenvolvido nesse trabalho visa o menor gasto de energia possível, essa opção foi rejeitada.

Em relação ao SmartConfig, o site da Espressif possui uma descrição a respeito:

The SmartConfig is a provisioning technology developed by TI to connect a new Wi-Fi device to a Wi-Fi network. It uses a mobile application to broadcast the network credentials from a smartphone, or a tablet, to an un-provisioned Wi-Fi device ([SmartConfig...](#)).

Podem ser encontrados vários aplicativos móveis na PlayStore que possuem a funcionalidade do SmartConfig. Caso se deseje criar a própria aplicação do SmartConfig, é possível encontrar repositórios no GitHub que implementam essa funcionalidade ([SIMONTOM/ESP32-SmartConfig-AndroidApp...](#),). Para não tornar o projeto muito complexo, foi escolhido utilizar uma aplicação da PlayStore, em vez de associar essa funcionalidade ao aplicativo de controlar a irrigação. Para acionar o serviço de conexão por SmartConfig, a biblioteca `ESP8266WiFi.h` disponibiliza a função mostrada em 3.1.

```
1 WiFi.beginSmartConfig()
```

Listing 3.1 – Função para acionar o serviço do SmartConfig

Para verificar se a conexão foi feita corretamente, é possível verificar o status da conexão pela função `WiFi.status()`. Caso ela retorne `WL_CONNECTED`, a conexão via SmartConfig foi feita com sucesso.

A função utilizada no código do microcontrolador para lidar com a conexão via SmartConfig é mostrada em 3.2.

```
1 bool IniciarSmartConfig() {
2     WiFi.mode(WIFI_STA); // Define o modo de estação
3     Serial.println("Iniciando SmartConfig...");
4     WiFi.beginSmartConfig(); // Inicia o modo SmartConfig
5
6     unsigned long tempo_inicial = millis();
7     while (millis() - tempo_inicial < WiFiSmartConfig_Timeout) {
8         if (WiFi.status() == WL_CONNECTED) {
```

```

9         // Se conectou, salva as novas credenciais
10        String currentSSID = WiFi.SSID();
11        String currentPW = WiFi.psk();
12        SalvarCredenciais(currentSSID, currentPW); // Salva no
13        LittleFS
14
15        Serial.println("Conectado pelo SmartConfig e credenciais
16        salvas!");
17        Serial.print("SSID conectado: ");
18        Serial.println(currentSSID);
19        Serial.print("PW conectado: ");
20        Serial.println(currentPW);
21        return true; // Conectado e salvo
22    }
23
24    delay(1000); // Espera 1 segundo para o SmartConfig ou conex?o
25    Serial.print(".");
26 }

```

Listing 3.2 – Função para configurar o SmartConfig

Para evitar que o microcontrolador fique buscando a conexão indefinidamente e prejudique a vida útil da bateria, a função retorna 0 depois de um tempo definido na variável `WiFiSmartConfig_Timeout`. Caso a conexão seja feita corretamente, a função retorna 1.

3.4.2 Armazenamento das credenciais do WiFi

Para proporcionar uma melhor experiência ao usuário, isentando-o de recorrer ao SmartConfig toda vez que o dispositivo é reiniciado, é necessário armazenar os dados de rede usados. Como opções para armazenar esses dados, é possível utilizar o `EEPROM.h`, que emula uma Electrically Erasable Programmable Read-Only Memory(EEPROM) ou o sistema de arquivos `LittleFS.h`.

Embora o microcontrolador ESP8266 não possua uma memória EEPROM, a primeira biblioteca, `EEPROM.h`, é capaz de emular o comportamento dessa memória. O site da STMicroelectronics explica a vantagem de se utilizar um emulador de EEPROM em vez de utilizar diretamente a memória flash:

Using flash memory to store long-term data can put additional stress on the flash memory, as erase operations are necessary to free up space but the EEPROM emulation library minimizes this stress by performing garbage collection operations to compact valid data and minimize free space. The library allows users to specify the maximum number of write operations before a garbage collection operation is performed, allowing

for control over the number of erase operations needed. ([EEPROM...](#), [2023](#)).

Para salvar os dados na biblioteca EEPROM, é necessário salvar individualmente cada caractere em uma posição da memória. Esse passo a passo torna o código maior, sendo mais suscetível a erros. Além disso, considerando que também será necessário utilizar a memória para salvar os dados dos alarmes, o gerenciamento de dois tipos de dados diferentes se tornaria mais complexo.

A segunda biblioteca, `LittleFS.h`, é um sistema de arquivos para utilizar a memória flash presente no ESP8266. Originalmente, o sistema de arquivos SPIFFS era utilizado, e possui as vantagens de um baixo overhead e baixo uso de memória RAM. Contudo esse sistema não suporta diretórios e consegue armazenar apenas arquivos pequenos. Atualmente, esse sistema está obsoleto, e não terá mais suporte futuramente. Em substituição a ele, surgiu o LittleFS, que possui melhor performance e suporte a diretórios, porém seu sistema de arquivos ocupa maior espaço e ele possui maior overhead. Contudo, considerando que neste trabalho o sistema de arquivos só deverá armazenar as credenciais de WiFi e os dados dos alarmes salvos, o uso do armazenamento não será uma preocupação. Por esses motivos, foi decidido utilizar a biblioteca `LittleFS.h`.

A função `SalvarCredenciais()`, utilizada no corpo da função `WiFiSmartConfig()` é utilizada para salvar o SSID e a senha contidos na struct `settings`. O código dessa função é mostrado em [3.3](#).

```

1 void SalvarCredenciais(const String& ssid, const String& password) {
2     File configFile = LittleFS.open(CONFIG_FILE, "w"); // "w" para
        sobrescrever o arquivo
3     if (!configFile) {
4         Serial.println("Erro ao criar/abrir arquivo para salvar
        credenciais.");
5         return;
6     }
7
8     configFile.println(ssid);
9     configFile.println(password);
10    configFile.close();
11    Serial.println("Credenciais salvas no LittleFS.");
12 }
```

Listing 3.3 – Função para salvar dados de login e senha do WiFi

A função `SalvarCredenciais()` abre um arquivo chamado `CONFIG_FILE` (definido como `"/wifi_config.txt"`) no sistema de arquivos LittleFS no modo de escrita (`"w"`). Se o arquivo for aberto com sucesso, o SSID e a senha são escritos cada um em uma nova linha usando `configFile.println()`. Após a escrita, o arquivo é fechado com `configFile.close()`.

Para recuperar os dados salvos na memória flash, foi criada a função `RecuperarCredenciaisWifi()`. Sua implementação é mostrada em [3.4](#).

```

1 bool RecuperarCredenciaisWiFi() {
2     File configFile = LittleFS.open(CONFIG_FILE, "r");
3     if (!configFile) {
4         Serial.println("Arquivo de configuração não encontrado ou erro
5         ao abrir.");
6         globalSSID = "";
7         globalPW = "";
8         return false;
9     }
10
11     globalSSID = configFile.readStringUntil('\n');
12     globalPW = configFile.readStringUntil('\n'); // Lê a segunda linha
13     para a senha
14     configFile.close();
15
16     // Remove espaços em branco ou caracteres de nova linha extras que
17     podem ser lidos
18     globalSSID.trim();
19     globalPW.trim();
20
21     Serial.print("SSID lido do LittleFS: ");
22     Serial.println(globalSSID);
23     Serial.print("PW lido do LittleFS: ");
24     Serial.println(globalPW);
25
26     if (globalSSID.length() > 0 && globalPW.length() > 0) {
27         return true;
28     } else {
29         Serial.println("Credenciais vazias ou inválidas no arquivo.");
30         return false;
31     }
32 }
```

Listing 3.4 – Função para recuperar os dados de login e senha do WiFi

A função `RecuperarCredenciaisWiFi()` tenta abrir o arquivo `CONFIG_FILE` no modo de leitura ("r"). Se o arquivo for encontrado e aberto com sucesso, as credenciais são lidas linha por linha usando `configFile.readStringUntil('\n')` e armazenadas nas variáveis globais `globalSSID` e `globalPW`. A função retorna true se as credenciais forem lidas e forem válidas (não vazias), e false caso contrário.

Após a primeira conexão através do SmartConfig, as credenciais do WiFi terão sido salvas na memória flash utilizando o LittleFS. Quando o microcontrolador reiniciar, ele

tentará se conectar à rede WiFi utilizando as últimas credenciais salvas. Isso é realizado pelo código mostrado em 3.5.

```

1 bool ConectarWiFiComCredenciaisSalvas() {
2     if (!RecuperarCredenciaisWiFi()) { // Tenta carregar as credenciais
3         Serial.println("Não foi possível carregar credenciais válidas do
4             LittleFS.");
5         return false;
6     }
7
8     Serial.print("Tentando conectar com SSID: ");
9     Serial.println(globalSSID);
10
11    // Conecta ao Wi-Fi
12    WiFi.begin(globalSSID.c_str(), globalPW.c_str());
13
14    unsigned long tempo_inicial = millis();
15    while (millis() - tempo_inicial < WiFiConnection_Timeout) {
16        if (WiFi.status() == WL_CONNECTED) {
17            Serial.println("\nConectado com credenciais salvas!");
18            return true; // Conectado com sucesso
19        }
20        delay(500); // Espera 0.5 segundo antes de verificar novamente
21        Serial.print(".");
22    }
23    Serial.println("\nTempo esgotado para conexão com credenciais salvas
24 .");
25    return false; // Tempo esgotado, não conectou
}

```

Listing 3.5 – Função para conectar ao WiFi com dados da memória flash

3.4.3 Criando um servidor

Para a criação do servidor que lidará com as requisições feitas pelo aplicativo, é possível utilizar duas bibliotecas do Arduino IDE: **WiFiServer** e **ESP8266WebServer**.

A primeira biblioteca citada possui um tratamento em baixo nível das requisições ao servidor. Com **WiFiServer**, o desenvolvedor precisa lidar diretamente com conexões do tipo **WiFiClient**, além de interpretar e responder manualmente às mensagens do protocolo HTTP. Isso pode ser útil em aplicações que demandam controle mais preciso sobre a comunicação, mas aumenta a complexidade do código. Suas funções principais incluem **begin()**, para iniciar o servidor, **available()**, para verificar se há clientes conectados, e métodos como **client.read()** e **client.print()** para ler e enviar dados ao cliente.

A segunda biblioteca é de mais alto nível, simplificando o desenvolvimento dos servidores, e não exigindo tanto conhecimento dos protocolos utilizados. Essa bibli-

teca consegue lidar com requisições HTTP como GET e POST ([ARDUINO/LIBRARIES/ESP8266WebServer...](#)), o que é suficiente para este trabalho. Embora essa biblioteca só consiga atender um cliente por vez, esta não é uma limitação relevante, considerando que não foi incluída nos requisitos não funcionais do projeto. Entre suas funções principais estão `begin()`, para iniciar o servidor, `handleClient()`, que processa requisições recebidas, e os métodos `on()` e `send()`, que permitem registrar rotas e enviar respostas HTTP com facilidade. Assim, o uso da `ESP8266WebServer` contribui para um desenvolvimento mais ágil e com menos propensão a erros, sendo a escolha adotada neste projeto.

No início do código, o servidor foi criado utilizando a porta 80, como mostrado em [3.6](#).

```
1   ESP8266WebServer server(80);
```

Listing 3.6 – Criação do servidor Web

Em seguida, no `void setup()`, foram definidos os endpoints do servidor através de `server.on()` e o servidor foi inicializado com `server.begin()` como mostrado em [3.7](#).

```
1   server.on("/status", status_sistema);
2   server.on("/config", configurar_timer);
3   server.on("/bateria", nivel_bateria);
4   server.begin();
```

Listing 3.7 – Criação dos endpoints e iniciação do servidor.

Após isso, no loop principal, foi executada a função da biblioteca `ESP8266WebServer.h` que verifica se há requisições de usuários. A função para realizar essa tarefa é mostrada em [3.8](#).

```
1   server.handleClient();
```

Listing 3.8 – Função para tratar requisições de clientes

3.4.4 Programação Over-The-Air

Para realizar a programação via WiFi, foi utilizada a biblioteca `ArduinoOTA`. Para realizar as configurações para a programação do módulo pelo OTA, foi criada a função `OTA_Initialization()`, mostrada em [3.9](#).

```
1 void OTA_Initialization() {
2     ArduinoOTA.setHostname("meutcc");
3     ArduinoOTA.setPassword("12345");
4     ArduinoOTA.onStart([]() {
5         String type;
6         if (ArduinoOTA.getCommand() == U_FLASH) {
7             type = "sketch";
8         } else { // U_SPIFFS
```

```

9         type = "filesystem";
10    }
11    Serial.println("Iniciando atualização " + type);
12    DesligarLeds();
13 });
14 ArduinoOTA.onEnd([]() {
15     Serial.println("\nFim da atualização!");
16     LigarVerde();
17 });
18 ArduinoOTA.onProgress([](unsigned int progress, unsigned int total)
19 {
20     Serial.printf("Progresso: %u%\r\n", (progress / (total / 100)));
21 });
22 ArduinoOTA.onError([](ota_error_t error) {
23     Serial.printf("Erro[%u]: ", error);
24     if (error == OTA_AUTH_ERROR) Serial.println("Autenticação falhou");
25     else if (error == OTA_BEGIN_ERROR) Serial.println("Erro ao iniciar");
26     else if (error == OTA_CONNECT_ERROR) Serial.println("Erro de conexão");
27     else if (error == OTA_RECEIVE_ERROR) Serial.println("Erro ao receber");
28     else if (error == OTA_END_ERROR) Serial.println("Erro ao finalizar");
29     // Sinalize um erro com os LEDs
30     ConnectionError(); // Ou uma função específica para erro OTA
31 });
32 ArduinoOTA.begin();
33 Serial.println("OTA inicializado. Aguardando atualizações...");
```

Listing 3.9 – Código para programação OTA

Para o funcionamento correto dessa funcionalidade, é necessário adicionar a função mostrada em [3.10](#) no loop principal.

```
1 ArduinoOTA.handle();
```

Listing 3.10 – Função para verificar programação por OTA

3.4.5 Monitoramento da bateria

Para fazer o monitoramento de quando é necessário trocar as pilhas do irrigador, foi criada uma função `EstadoBateria()`. Seu código é mostrado em [3.11](#).

```
1 BatteryState EstadoBateria(){
```

```

2     int leitura_bateria = analogRead(PINO_ADC_BATERIA);
3     float battery_voltage = (((float) leitura_bateria)/1023.0)*
4         FATOR_DIVISOR;
5     if(battery_voltage > VOLTAGEM_ALTA)
6     {
7         return ALTO;
8     }
9     else{
10        if(battery_voltage > VOLTAGEM_BAIXA)
11            return MEDIO;
12        else
13            return BAIXO;
14    }

```

Listing 3.11 – Função para retorno sobre estado da bateria

O tipo BatteryState foi criado através de uma `enum`, como mostrado em [3.12](#).

```

1 enum BatteryState{
2     ALTO,
3     MEDIO,
4     BAIXO,
5 };

```

Listing 3.12 – Definição do tipo BatteryState

As variáveis utilizadas no código foram definidas no início do código como mostrado em [3.13](#):

```

1 const int PINO_ADC_BATERIA = A0;
2 const float R1 = 1000000.0;
3 const float R2 = 470000.0;
4 const float FATOR_DIVISOR = (R1 + R2) / R2;
5 const float VOLTAGEM_ALTA = 2.80;
6 const float VOLTAGEM_BAIXA = 2.30;

```

Listing 3.13 – Definição das variáveis no início do código

3.5 Desenvolvimento do aplicativo

3.5.1 Persistência das Configurações

A fim de manter as configurações criadas pelo usuário salvas, mesmo após o fechamento do aplicativo, optou-se por utilizar um banco de dados no aplicativo, através da base de dados Room. Para o funcionamento correto dessa biblioteca, as linhas de código mostradas em `??` devem ser adicionadas ao arquivo `build.gradle.kts`.

```

1  plugins{
2    id("kotlin-kapt")
3    id("com.google.devtools.ksp")
4  }
5  dependencies{
6    implementation(libs.androidx.room.runtime)
7    annotationProcessor(libs.androidx.room.compiler)
8    ksp(libs.androidx.room.compiler.v252)
9    implementation(libs.androidx.room.ktx)
10   implementation(libs.androidx.lifecycle.livedata.ktx)
11   implementation(libs.androidx.lifecycle.runtime.ktx.v261)
12 }
```

Os códigos 3.14, 3.15 e 3.16 mostram os arquivos para a criação e definição do banco de dados.

```

1  @Database(
2    entities = [Configuration::class],
3    version = 1
4 )
5 abstract class BaseDados: RoomDatabase(){
6   abstract val ConfigurationDao: ConfigurationDao
7
8   companion object {
9     @Volatile
10    private var INSTANCE: BaseDados? = null
11
12    fun getDatabase(context: Context): BaseDados {
13      return INSTANCE ?: synchronized(this) {
14        val instance = Room.databaseBuilder(
15          context.applicationContext,
16          BaseDados::class.java,
17          "database"
18        ).build()
19        INSTANCE = instance
20        instance
21      }
22    }
23  }
24 }
```

Listing 3.14 – Arquivo Dados_Base.kt

```

1  @Dao
2 interface ConfigurationDao{
3   @Query("SELECT * FROM configuration")
4   fun getAll(): Flow<List<Configuration>>
5
6   @Upsert
```

```

7     suspend fun upsert(num: Configuration)
8
9     @Delete
10    suspend fun delete(num: Configuration)
11
12    @Query("DELETE FROM configuration")
13    suspend fun deleteAll()
14
15    @Query("SELECT * FROM Configuration WHERE id = :id")
16    suspend fun getItemById(id: Int): Configuration?
17
18 }

```

Listing 3.15 – Arquivo DataAccess.kt

```

1 @Entity
2 data class Configuration(
3     @PrimaryKey(autoGenerate = true) val id: Int=0,
4     @ColumnInfo(name = "Nome do alarme") val nomeAlarme: String,
5     @ColumnInfo(name = "Ativação") val ativo: Boolean,
6     @ColumnInfo(name = "Dias da Semana") val diasSemana: String
7 )

```

Listing 3.16 – Arquivo EntidadeDados.kt

Foi criado também um arquivo de repositório para servir de interface do banco de dados com a ViewModel. Seu código é mostrado em 3.17.

```

1   class Repository(private val Configuration_dao : ConfigurationDao){
2       val allItems : Flow<List<Configuration>> = Configuration_dao.getAll()
3
4       suspend fun insert(num: Configuration) {
5           Configuration_dao.upsert(num)
6       }
7
8       suspend fun delete(num: Configuration) {
9           Configuration_dao.delete(num)
10      }
11
12      suspend fun getItemById(id: Int): Configuration? {
13          return Configuration_dao.getItemById(id)
14      }
15
16      suspend fun deleteAll(){
17          Configuration_dao.deleteAll()
18      }
19 }

```

Listing 3.17 – Arquivo Repo.kt

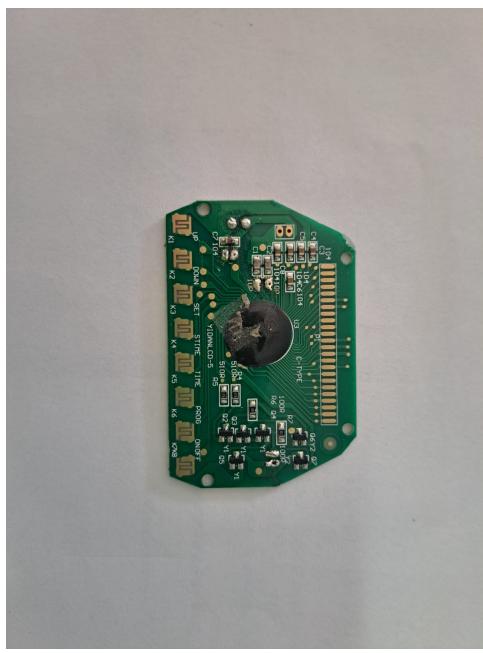
3.6 Montagem

Para a montagem dos componentes e sua fixação na placa, foi escolhido utilizar uma placa de fenolite perfurada, devido ao seu baixo preço e fácil acesso. Apesar disso, a fabricação de uma PCB específica para esse projeto é uma melhor escolha a longo prazo, devido à maior robustez e à possibilidade de utilizar menos espaço. Contudo, devido a questões como o tempo para entrega, preço de fabricação e quantidade mínima para produção, a fabricação de uma PCB foi descartada nesse momento.

3.6.1 Irrigador Trato

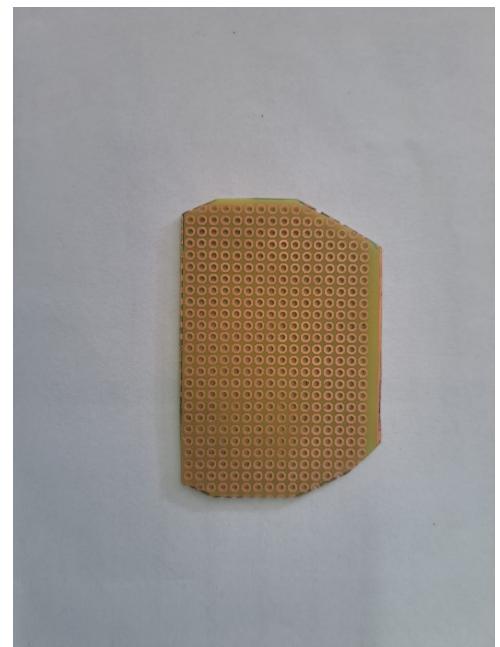
Em relação ao irrigador da marca Trato, o espaço para montagem da placa é bastante reduzido, sendo limitado superiormente pelo local de parafusamento da placa e inferiormente pelo local dos botões. Em razão disso, a fim de se aproveitar o maior espaço possível, optou-se por utilizar a placa original da Trato como referência para o corte da placa de fenolite. A imagem 6 mostra a placa original do irrigador da Trato e a imagem 7 mostra a placa de fenolite cortada com os moldes da original.

Figura 6 – Placa original da Trato



Fonte: Autoria própria

Figura 7 – Placa de fenolite utilizada



Fonte: Autoria própria

3.6.2 Irrigador Amanco

Da mesma forma que no irrigador Trato, a PCB original do irrigador da Amanco foi utilizada como molde da placa de fenolite para montagem dos componentes. Alguns

locais onde a placa antiga era parafusada também foram aproveitados para a fixação da nova placa.

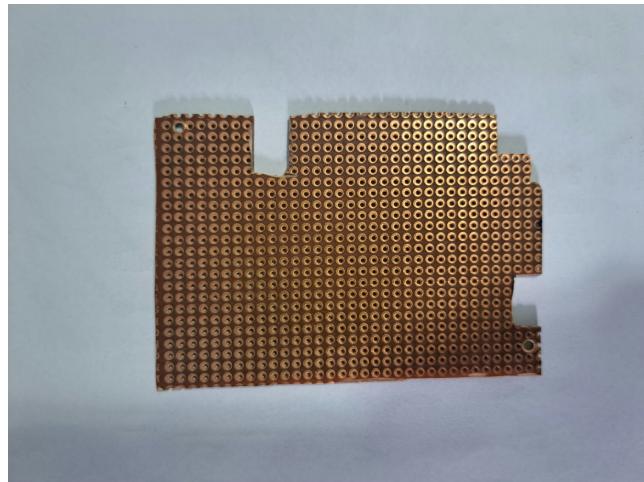
A placa utilizada pelo irrigador da Amanco é mostrada na figura 8. A figura 9 mostra a placa de fenolite utilizada no projeto para substituir a placa antiga.

Figura 8 – Placa original da Amanco



Fonte: Autoria própria

Figura 9 – Placa de fenolite utilizada



Fonte: Autoria própria

3.7 Comentários

Uma das restrições importantes encontradas nesse projeto foi a questão do espaço disponível para inserir os componentes. Caso houvesse mais espaço disponível, seria interessante adicionar um RTC externo, para despertar o módulo com mais precisão de horário.

4 Resultados e Discussão

4.1 Vestibulum ante ipsum primis

Nulla non mauris vitae wisi posuere convallis. Sed eu nulla nec eros scelerisque pharetra. Nullam varius. Etiam dignissim elementum metus. Vestibulum faucibus, metus sit amet mattis rhoncus, sapien dui laoreet odio, nec ultricies nibh augue a enim. Fusce in ligula. Quisque at magna et nulla commodo consequat. Proin accumsan imperdiet sem. Nunc porta. Donec feugiat mi at justo. Phasellus facilisis ipsum quis ante. In ac elit eget ipsum pharetra faucibus. Maecenas viverra nulla in massa.

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo.

Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

4.2 Integer porta neque vitae massa

Maecenas non massa. Vestibulum pharetra nulla at lorem. Duis quis quam id lacus dapibus interdum. Nulla lorem. Donec ut ante quis dolor bibendum condimentum. Etiam egestas tortor vitae lacus. Praesent cursus. Mauris bibendum pede at elit. Morbi et felis a lectus interdum facilisis. Sed suscipit gravida turpis. Nulla at lectus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent nonummy luctus nibh. Proin turpis nunc, congue eu, egestas ut, fringilla at, tellus. In hac habitasse platea dictumst.

Vivamus eu tellus sed tellus consequat suscipit. Nam orci orci, malesuada id, gravida nec, ultricies vitae, erat. Donec risus turpis, luctus sit amet, interdum quis, porta sed, ipsum. Suspendisse condimentum, tortor at egestas posuere, neque metus tempor orci, et tincidunt urna nunc a purus. Sed facilisis blandit tellus. Nunc risus sem, suscipit nec, eleifend quis, cursus quis, libero. Curabitur et dolor. Sed vitae sem. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas ante. Duis ullamcorper enim. Donec tristique enim eu leo. Nullam molestie elit eu dolor. Nullam bibendum, turpis vitae tristique gravida, quam sapien tempor lectus, quis pretium tellus purus ac quam. Nulla facilisi.

Duis aliquet dui in est. Donec eget est. Nunc lectus odio, varius at, fermentum in, accumsan non, enim. Aliquam erat volutpat. Proin sit amet nulla ut eros consectetur cursus. Phasellus dapibus aliquam justo. Nunc laoreet. Donec consequat placerat magna. Duis pretium tincidunt justo. Sed sollicitudin vestibulum quam. Nam quis ligula. Vivamus at metus. Etiam imperdiet imperdiet pede. Aenean turpis. Fusce augue velit, scelerisque sollicitudin, dictum vitae, tempor et, pede. Donec wisi sapien, feugiat in, fermentum ut, sollicitudin adipiscing, metus.

Donec vel nibh ut felis consectetur laoreet. Donec pede. Sed id quam id wisi laoreet suscipit. Nulla lectus dolor, aliquam ac, fringilla eget, mollis ut, orci. In pellentesque justo in ligula. Maecenas turpis. Donec eleifend leo at felis tincidunt consequat. Aenean turpis metus, malesuada sed, condimentum sit amet, auctor a, wisi. Pellentesque sapien elit, bibendum ac, posuere et, congue eu, felis. Vestibulum mattis libero quis metus scelerisque ultrices. Sed purus.

Donec molestie, magna ut luctus ultrices, tellus arcu nonummy velit, sit amet pulvinar elit justo et mauris. In pede. Maecenas euismod elit eu erat. Aliquam augue wisi, facilisis congue, suscipit in, adipiscing et, ante. In justo. Cras lobortis neque ac ipsum. Nunc fermentum massa at ante. Donec orci tortor, egestas sit amet, ultrices eget,

venenatis eget, mi. Maecenas vehicula leo semper est. Mauris vel metus. Aliquam erat volutpat. In rhoncus sapien ac tellus. Pellentesque ligula.

Cras dapibus, augue quis scelerisque ultricies, felis dolor placerat sem, id porta velit odio eu elit. Aenean interdum nibh sed wisi. Praesent sollicitudin vulputate dui. Praesent iaculis viverra augue. Quisque in libero. Aenean gravida lorem vitae sem ullamcorper cursus. Nunc adipiscing rutrum ante. Nunc ipsum massa, faucibus sit amet, viverra vel, elementum semper, orci. Cras eros sem, vulputate et, tincidunt id, ultrices eget, magna. Nulla varius ornare odio. Donec accumsan mauris sit amet augue. Sed ligula lacus, laoreet non, aliquam sit amet, iaculis tempor, lorem. Suspendisse eros. Nam porta, leo sed congue tempor, felis est ultrices eros, id mattis velit felis non metus. Curabitur vitae elit non mauris varius pretium. Aenean lacus sem, tincidunt ut, consequat quis, porta vitae, turpis. Nullam laoreet fermentum urna. Proin iaculis lectus.

5 Conclusões

5.1 Pellentesque sit amet pede ac sem eleifend

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

Sed consequat tellus et tortor. Ut tempor laoreet quam. Nullam id wisi a libero tristique semper. Nullam nisl massa, rutrum ut, egestas semper, mollis id, leo. Nulla ac massa eu risus blandit mattis. Mauris ut nunc. In hac habitasse platea dictumst. Aliquam eget tortor. Quisque dapibus pede in erat. Nunc enim. In dui nulla, commodo at, consectetur nec, malesuada nec, elit. Aliquam ornare tellus eu urna. Sed nec metus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

5.2 Phasellus id magna

Sed eleifend, eros sit amet faucibus elementum, urna sapien consectetur mauris, quis egestas leo justo non risus. Morbi non felis ac libero vulputate fringilla. Mauris libero eros, lacinia non, sodales quis, dapibus porttitor, pede. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi dapibus mauris condimentum nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Etiam sit amet erat. Nulla varius. Etiam tincidunt dui vitae turpis. Donec leo. Morbi vulputate convallis est. Integer aliquet. Pellentesque aliquet sodales urna.

Nullam eleifend justo in nisl. In hac habitasse platea dictumst. Morbi nonummy. Aliquam ut felis. In velit leo, dictum vitae, posuere id, vulputate nec, ante. Maecenas vitae pede nec dui dignissim suscipit. Morbi magna. Vestibulum id purus eget velit laoreet laoreet. Praesent sed leo vel nibh convallis blandit. Ut rutrum. Donec nibh. Donec interdum. Fusce sed pede sit amet elit rhoncus ultrices. Nullam at enim vitae pede vehicula iaculis.

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Aenean nonummy turpis id odio. Integer euismod imperdiet turpis. Ut nec leo nec diam imperdiet lacinia. Etiam eget lacus eget mi ultricies posuere. In placerat tristique tortor. Sed porta vestibulum metus. Nulla iaculis sollicitudin pede. Fusce luctus tellus in dolor. Curabitur auctor velit a sem. Morbi sapien. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Donec adipiscing urna vehicula nunc. Sed ornare leo in leo. In rhoncus leo ut dui. Aenean dolor quam, volutpat nec, fringilla id, consectetur vel, pede.

Nulla malesuada risus ut urna. Aenean pretium velit sit amet metus. Duis iaculis. In hac habitasse platea dictumst. Nullam molestie turpis eget nisl. Duis a massa id pede dapibus ultricies. Sed eu leo. In at mauris sit amet tortor bibendum varius. Phasellus justo risus, posuere in, sagittis ac, varius vel, tortor. Quisque id enim. Phasellus consequat, libero pretium nonummy fringilla, tortor lacus vestibulum nunc, ut rhoncus ligula neque id justo. Nullam accumsan euismod nunc. Proin vitae ipsum ac metus dictum tempus. Nam ut wisi. Quisque tortor felis, interdum ac, sodales a, semper a, sem. Curabitur in velit sit amet dui tristique sodales. Vivamus mauris pede, lacinia eget, pellentesque quis, scelerisque eu, est. Aliquam risus. Quisque bibendum pede eu dolor.

Donec tempus neque vitae est. Aenean egestas odio sed risus ullamcorper ullamcorper. Sed in nulla a tortor tincidunt egestas. Nam sapien tortor, elementum sit amet, aliquam in, porttitor faucibus, enim. Nullam congue suscipit nibh. Quisque convallis. Praesent arcu nibh, vehicula eget, accumsan eu, tincidunt a, nibh. Suspendisse vulputate, tortor quis adipiscing viverra, lacus nibh dignissim tellus, eu suscipit risus ante fringilla diam. Quisque a libero vel pede imperdiet aliquet. Pellentesque nunc nibh, eleifend a, consequat consequat, hendrerit nec, diam. Sed urna. Maecenas laoreet eleifend neque. Vivamus purus odio, eleifend non, iaculis a, ultrices sit amet, urna. Mauris faucibus odio vitae risus. In nisl. Praesent purus. Integer iaculis, sem eu egestas lacinia, lacus pede scelerisque augue, in ullamcorper dolor eros ac lacus. Nunc in libero.

Fusce suscipit cursus sem. Vivamus risus mi, egestas ac, imperdiet varius, faucibus quis, leo. Aenean tincidunt. Donec suscipit. Cras id justo quis nibh scelerisque dignissim. Aliquam sagittis elementum dolor. Aenean consectetur justo in pede. Curabitur ullamcorper ligula nec orci. Aliquam purus turpis, aliquam id, ornare vitae, porttitor non, wisi. Maecenas luctus porta lorem. Donec vitae ligula eu ante pretium varius. Proin tortor metus, convallis et, hendrerit non, scelerisque in, urna. Cras quis libero eu ligula bibendum tempor. Vivamus tellus quam, malesuada eu, tempus sed, tempor sed, velit. Donec lacinia auctor libero.

Praesent sed neque id pede mollis rutrum. Vestibulum iaculis risus. Pellentesque lacus. Ut quis nunc sed odio malesuada egestas. Duis a magna sit amet ligula tristique pretium. Ut pharetra. Vestibulum imperdiet magna nec wisi. Mauris convallis. Sed accu-

msan sollicitudin massa. Sed id enim. Nunc pede enim, lacinia ut, pulvinar quis, suscipit semper, elit. Cras accumsan erat vitae enim. Cras sollicitudin. Vestibulum rutrum blandit massa.

Sed gravida lectus ut purus. Morbi laoreet magna. Pellentesque eu wisi. Proin turpis. Integer sollicitudin augue nec dui. Fusce lectus. Vivamus faucibus nulla nec lacus. Integer diam. Pellentesque sodales, enim feugiat cursus volutpat, sem mauris dignissim mauris, quis consequat sem est fermentum ligula. Nullam justo lectus, condimentum sit amet, posuere a, fringilla mollis, felis. Morbi nulla nibh, pellentesque at, nonummy eu, sollicitudin nec, ipsum. Cras neque. Nunc augue. Nullam vitae quam id quam pulvinar blandit. Nunc sit amet orci. Aliquam erat elit, pharetra nec, aliquet a, gravida in, mi. Quisque urna enim, viverra quis, suscipit quis, tincidunt ut, sapien. Cras placerat consequat sem. Curabitur ac diam. Curabitur diam tortor, mollis et, viverra ac, tempus vel, metus.

Referências

ABOUT Upcycling. 2017. Disponível em: <<https://upcyclethat.com/about-upcycling/>>. Citado na página 23.

ARDUINO/LIBRARIES/ESP8266WebServer at master · esp8266/Arduino. Disponível em: <<https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WebServer>>. Citado na página 44.

BREAKOUT PCB for ESP8266 WIFI ESP-07 ESP-12. Disponível em: <<https://www.sunrom.com/p/breakout-pcb-for-esp8266-wifi-esp-07-esp-12>>. Citado na página 35.

CONSTRUINDO um ESP32 de automação residencial com servidor web e EEPROM. 2025. Section: ESP32 Projects. Disponível em: <<https://iotcircuithub.com/esp32-home-automation-with-web-server-eeprom>>. Citado na página 31.

CREATE an Android App with Android Studio to control an LED over WiFi using NodeMCU. Disponível em: <<https://iotdesignpro.com/projects/create-android-app-with-android-studio-to-control-led-over-wifi-using-nodemcu>>. Citado na página 30.

EDISON SCIENCE CORNER. *Arduino IoT Cloud Based Battery Monitor / Arduino IoT*. 2023. Disponível em: <<https://www.youtube.com/watch?v=eNFwGSiGeyw>>. Citado na página 32.

Education is Life (joed goh). *04 Easy IoT project w/ Arduino IoT Cloud - ESP32 / DHT11 Humidity & Temperature / Relay & DC Motor*. 2022. Disponível em: <<https://www.youtube.com/watch?v=kcMzBV3XW2E>>. Citado na página 27.

Education is Life (joed goh). *06. ESP32 + Firebase Realtime Database + Android Studio - IoT Development*. 2022. Disponível em: <<https://www.youtube.com/watch?v=LaUzGdtLFiQ>>. Citado na página 29.

EEPROM emulation: working principle. 2023. Section: STM32 MCUs Products. Disponível em: <<https://community.st.com/t5/stm32-mcus-products/eeprom-emulation-working-principle/td-p/613628>>. Citado na página 41.

ELSYS.COM. *Painel de LCD e sol. Um não foi feito para o outro*. 2018. Disponível em: <<https://elsys.com/blog/painel-de-lcd-e-sol-um-nao-foi-feito-para-o-outro/>>. Citado na página 24.

ESP32 Matter Protocol control Multiple Relays - 2025. 2024. Section: ESP32 Projects. Disponível em: <<https://iotcircuithub.com/esp32-matter-protocol-control-relays>>. Citado na página 29.

ESP32 Web Server + ESP-NOW Relay control with OLED 2025. 2025. Section: ESP32 Projects. Disponível em: <<https://iotcircuithub.com/esp32-web-server-esp-now-relay-control-with-oled>>. Citado na página 31.

FILHO, W. de P. P. *Engenharia de software: fundamentos, métodos e padrões*. Grupo Gen - LTC, 2009. ISBN 978-85-216-1650-4. Disponível em: <<https://books.google.com.br/books?id=mQ4XPwAACAAJ>>. Citado na página 33.

FREITAS, G. M. d. *ESP8266 Salvando Credenciais Wi-Fi na EEPROM Através de um Access Point (AP)*. 2021. Disponível em: <<https://blog.smartkits.com.br/esp8266-salvando-credenciais-wi-fi-na-eeprom-atraves-de-um-access-point-ap/>>. Citado na página 39.

GET started with Matter Protocol using ESP ZeroCode. 2023. Section: ESP32 Projects. Disponível em: <<https://iotcircuithub.com/matter-protocol-using-esp-zerocode/>>. Citado na página 29.

GPIO Maximum current Imax - ESP8266 Developer Zone. Disponível em: <<http://bbs.espressif.com/viewtopic.php?t=139>>. Citado na página 36.

HOFFMANN, F. *Débora Garofalo*. Disponível em: <<https://deboragarofalo.com.br/robotica-com-sucata/>>. Citado na página 27.

How to drive a DC motor without a motor driver module. Disponível em: <<https://techexplorations.com/guides/arduino/motors/dc-motor-with-transistor>>. Citado 2 vezes nas páginas 32 e 36.

Indrek. *Programming ESP-12E / ESP-12F / NodeMCU With Arduino IDE / Step by Step Guide*. 2020. Disponível em: <https://www.youtube.com/watch?v=_iX67plFeLs>. Citado na página 33.

IoT Frontier. *Azure IoT Project | Connect Raspberry Pi and Ultrasonic Sensor with Azure IoT Hub*. 2023. Disponível em: <<https://www.youtube.com/watch?v=45vG1MZOJPM>>. Citado 2 vezes nas páginas 27 e 29.

IoT Frontier. *ESP32 IoT Project with ThingSpeak using Wokwi Simulator*. 2024. Disponível em: <<https://www.youtube.com/watch?v=I818NJD-7Is>>. Citado na página 30.

IoT Projects Ideas. *ESP8266 Monitor its Own Battery Level using Blynk IoT for Battery Powered Projects*. 2022. Disponível em: <<https://www.youtube.com/watch?v=Bxc2HnGHMyg>>. Citado 3 vezes nas páginas 27, 31 e 32.

L298 Dual H-Bridge Motor Driver Datasheet. 2025. Section: Motor Driver Datasheets. Disponível em: <<https://www.st.com/resource/en/datasheet/l298.pdf>>. Citado na página 37.

MARSHALL, I. *ESP8266 - Breakout!* 2017. Disponível em: <<https://smallbits.marshall-tribe.net/blog/2017/02/23/esp8266-breakout>>. Citado na página 35.

MINI Motor DC 3-6V. Disponível em: <<https://www.eletrogate.com/mini-motor-dc>>. Citado na página 36.

MINI Ponte H Dupla L298N. Disponível em: <<https://www.eletrogate.com/mini-ponte-h-dupla-l298n>>. Citado na página 38.

- OLIVEIRA, M. *Meta-Circuitos: democratizando o ensino maker*. 2019. Section: Educação. Disponível em: <<https://roteirobaby.com.br/2019/12/meta-circuitos-democratizando-o-ensino-maker.html>>. Citado na página 27.
- PROGRAMMING an ESP8266 (ESP-12x) via a USB to Serial adaptor. 2017. Disponível em: <<https://www.b4x.com/android/forum/threads/programming-an-esp8266-esp-12x-via-a-usb-to-serial-adaptor.79177/>>. Citado na página 33.
- RECICLAGEM. Disponível em: <<https://antigo.mma.gov.br/informma/item/7656-reciclagem.html>>. Citado na página 23.
- RENE, E. R. et al. Electronic waste generation, recycling and resource recovery: Technological perspectives and trends. *Journal of Hazardous Materials*, v. 416, p. 125664, ago. 2021. ISSN 0304-3894. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0304389421006282>>. Citado na página 23.
- ROBOCORE. *Irrigador Automático com Arduino*. Disponível em: <<https://www.robocore.net/tutoriais/334>>. Citado na página 32.
- Robu.in. *How to Send Data from ESP32 to ThingSpeak: Complete IoT Project Guide*. 2024. Disponível em: <<https://www.youtube.com/watch?v=Pr-A4f05XEs>>. Citado na página 30.
- SADDAM. *Arduino based Automatic Plant Irrigation System with Message Alert*. 2016. Disponível em: <<https://circuitdigest.com/microcontroller-projects/arduino-automatic-plant-watering-system>>. Citado 3 vezes nas páginas 24, 27 e 32.
- SANTOS, R. *ESP8266 NodeMCU MQTT Publish BME680 Sensor Readings (Arduino) / Random Nerd Tutorials*. 2020. Disponível em: <<https://randomnerdtutorials.com/esp8266-nodemcu-mqtt-publish-bme680-arduino/>>. Citado na página 29.
- SANTOS, R. *ESP32 Web Server: DC Motor (Arduino IDE) / Random Nerd Tutorials*. 2024. Disponível em: <<https://randomnerdtutorials.com/esp32-web-server-dc-motor-arduino/>>. Citado 3 vezes nas páginas 24, 31 e 32.
- SIMONTOM/ESP32-SmartConfig-AndroidApp: Shows how e-z it is to use Esptouch SmartConfig (Android app got from <https://github.com/EspressifApp/EspTouchForAndroid>)). Disponível em: <<https://github.com/simontom/ESP32-SmartConfig-AndroidApp/tree/master>>. Citado na página 39.
- SmartConfig ESP32 ESP-IDF Programming Guide v5.4.1 documentation. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/network/esp__smartconfig.html>. Citado na página 39.
- SONY. [Notebook] Observações sobre a tela LCD / Sony Brasil. 2025. Disponível em: <<https://www.sony.com.br/electronics/support/articles/00097792>>. Citado na página 24.
- SriTu Hobby. *How to make a plant watering system with ESP32 board and Blynk app #sritu_hobby #esp32project*. 2023. Disponível em: <<https://www.youtube.com/watch?v=Ix3a3ThsHfA>>. Citado na página 24.

TIP122 Darlington Transistor Datasheet. 2025. Section: Transistor Datasheets. Disponível em: <<https://www.onsemi.com/pdf/datasheet/tip122-d.pdf>>. Citado na página 36.

TRATO Temporizador Irrigaçao Automático para Hortas e Jardins | Amazon.com.br. Disponível em: <<https://www.amazon.com.br/Temporizador-Irriga%C3%A7%C3%A3o-Program%C3%A1vel-Autom%C3%A1tico-Jardins/dp/B09NC7MQ24>>. Citado na página 23.

witnessmenow. *3 Simple Ways of Programming an ESP8266 12X Module*. Disponível em: <<https://www.instructables.com/3-Simple-Ways-of-Programming-an-ESP8266-12X-Module/>>. Citado na página 33.

Apêndices

APÊNDICE A – Quisque libero justo

Quisque facilisis auctor sapien. Pellentesque gravida hendrerit lectus. Mauris rutrum sodales sapien. Fusce hendrerit sem vel lorem. Integer pellentesque massa vel augue. Integer elit tortor, feugiat quis, sagittis et, ornare non, lacus. Vestibulum posuere pellentesque eros. Quisque venenatis ipsum dictum nulla. Aliquam quis quam non metus eleifend interdum. Nam eget sapien ac mauris malesuada adipiscing. Etiam eleifend neque sed quam. Nulla facilisi. Proin a ligula. Sed id dui eu nibh egestas tincidunt. Suspendisse arcu.

Maecenas dui. Aliquam volutpat auctor lorem. Cras placerat est vitae lectus. Curabitur massa lectus, rutrum euismod, dignissim ut, dapibus a, odio. Ut eros erat, vulputate ut, interdum non, porta eu, erat. Cras fermentum, felis in porta congue, velit leo facilisis odio, vitae consectetur lorem quam vitae orci. Sed ultrices, pede eu placerat auctor, ante ligula rutrum tellus, vel posuere nibh lacus nec nibh. Maecenas laoreet dolor at enim. Donec molestie dolor nec metus. Vestibulum libero. Sed quis erat. Sed tristique. Duis pede leo, fermentum quis, consectetur eget, vulputate sit amet, erat.

Donec vitae velit. Suspendisse porta fermentum mauris. Ut vel nunc non mauris pharetra varius. Duis consequat libero quis urna. Maecenas at ante. Vivamus varius, wisi sed egestas tristique, odio wisi luctus nulla, lobortis dictum dolor ligula in lacus. Vivamus aliquam, urna sed interdum porttitor, metus orci interdum odio, sit amet euismod lectus felis et leo. Praesent ac wisi. Nam suscipit vestibulum sem. Praesent eu ipsum vitae pede cursus venenatis. Duis sed odio. Vestibulum eleifend. Nulla ut massa. Proin rutrum mattis sapien. Curabitur dictum gravida ante.

Phasellus placerat vulputate quam. Maecenas at tellus. Pellentesque neque diam, dignissim ac, venenatis vitae, consequat ut, lacus. Nam nibh. Vestibulum fringilla arcu mollis arcu. Sed et turpis. Donec sem tellus, volutpat et, varius eu, commodo sed, lectus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque enim arcu, suscipit nec, tempus at, imperdiet vel, metus. Morbi volutpat purus at erat. Donec dignissim, sem id semper tempus, nibh massa eleifend turpis, sed pellentesque wisi purus sed libero. Nullam lobortis tortor vel risus. Pellentesque consequat nulla eu tellus. Donec velit. Aliquam fermentum, wisi ac rhoncus iaculis, tellus nunc malesuada orci, quis volutpat dui magna id mi. Nunc vel ante. Duis vitae lacus. Cras nec ipsum.

Morbi nunc. Aliquam consectetur varius nulla. Phasellus eros. Cras dapibus porttitor risus. Maecenas ultrices mi sed diam. Praesent gravida velit at elit vehicula porttitor. Phasellus nisl mi, sagittis ac, pulvinar id, gravida sit amet, erat. Vestibulum est. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur id sem elementum leo ru-

trum hendrerit. Ut at mi. Donec tincidunt faucibus massa. Sed turpis quam, sollicitudin a, hendrerit eget, pretium ut, nisl. Duis hendrerit ligula. Nunc pulvinar congue urna.

APÊNDICE B – Nullam elementum urna vel imperdiet sodales elit ipsum pharetra

Nunc velit. Nullam elit sapien, eleifend eu, commodo nec, semper sit amet, elit. Nulla lectus risus, condimentum ut, laoreet eget, viverra nec, odio. Proin lobortis. Curabitur dictum arcu vel wisi. Cras id nulla venenatis tortor congue ultrices. Pellentesque eget pede. Sed eleifend sagittis elit. Nam sed tellus sit amet lectus ullamcorper tristique. Mauris enim sem, tristique eu, accumsan at, scelerisque vulputate, neque. Quisque lacus. Donec et ipsum sit amet elit nonummy aliquet. Sed viverra nisl at sem. Nam diam. Mauris ut dolor. Curabitur ornare tortor cursus velit.

Morbi tincidunt posuere arcu. Cras venenatis est vitae dolor. Vivamus scelerisque semper mi. Donec ipsum arcu, consequat scelerisque, viverra id, dictum at, metus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut pede sem, tempus ut, porttitor bibendum, molestie eu, elit. Suspendisse potenti. Sed id lectus sit amet purus faucibus vehicula. Praesent sed sem non dui pharetra interdum. Nam viverra ultrices magna.

Aenean laoreet aliquam orci. Nunc interdum elementum urna. Quisque erat. Nullam tempor neque. Maecenas velit nibh, scelerisque a, consequat ut, viverra in, enim. Duis magna. Donec odio neque, tristique et, tincidunt eu, rhoncus ac, nunc. Mauris malesuada malesuada elit. Etiam lacus mauris, pretium vel, blandit in, ultricies id, libero. Phasellus bibendum erat ut diam. In congue imperdiet lectus.

Aenean scelerisque. Fusce pretium porttitor lorem. In hac habitasse platea dictumst. Nulla sit amet nisl at sapien egestas pretium. Nunc non tellus. Vivamus aliquet. Nam adipiscing euismod dolor. Aliquam erat volutpat. Nulla ut ipsum. Quisque tincidunt auctor augue. Nunc imperdiet ipsum eget elit. Aliquam quam leo, consectetur non, ornare sit amet, tristique quis, felis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque interdum quam sit amet mi. Pellentesque mauris dui, dictum a, adipiscing ac, fermentum sit amet, lorem.

Ut quis wisi. Praesent quis massa. Vivamus egestas risus eget lacus. Nunc tincidunt, risus quis bibendum facilisis, lorem purus rutrum neque, nec porta tortor urna quis orci. Aenean aliquet, libero semper volutpat luctus, pede erat lacinia augue, quis rutrum sem ipsum sit amet pede. Vestibulum aliquet, nibh sed iaculis sagittis, odio dolor blandit augue, eget mollis urna tellus id tellus. Aenean aliquet aliquam nunc. Nulla ultricies justo eget orci. Phasellus tristique fermentum leo. Sed massa metus, sagittis ut, semper ut, pharetra vel, erat. Aliquam quam turpis, egestas vel, elementum in, egestas sit amet, lorem. Duis convallis, wisi sit amet mollis molestie, libero mauris porta dui, vitae aliquam

arcu turpis ac sem. Aliquam aliquet dapibus metus.

Anexos

ANEXO A – Morbi ultrices rutrum lorem

Vivamus commodo eros eleifend dui. Vestibulum in leo eu erat tristique mattis. Cras at elit. Cras pellentesque. Nullam id lacus sit amet libero aliquet hendrerit. Proin placerat, mi non elementum laoreet, eros elit tincidunt magna, a rhoncus sem arcu id odio. Nulla eget leo a leo egestas facilisis. Curabitur quis velit. Phasellus aliquam, tortor nec ornare rhoncus, purus urna posuere velit, et commodo risus tellus quis tellus. Vivamus leo turpis, tempus sit amet, tristique vitae, laoreet quis, odio. Proin scelerisque bibendum ipsum. Etiam nisl. Praesent vel dolor. Pellentesque vel magna. Curabitur urna. Vivamus congue urna in velit. Etiam ullamcorper elementum dui. Praesent non urna. Sed place-
rat quam non mi. Pellentesque diam magna, ultricies eget, ultrices placerat, adipisc-
ing rutrum, sem.

ANEXO B – Cras non urna sed feugiat cum sociis natoque penatibus

Morbi sem. Nulla facilisi. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nulla facilisi. Morbi sagittis ultrices libero. Praesent eu ligula sed sapien auctor sagittis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Donec vel nunc. Nunc fermentum, lacus id aliquam porta, dui tortor euismod eros, vel molestie ipsum purus eu lacus. Vivamus pede arcu, euismod ac, tempus id, pretium et, lacus. Curabitur sodales dapibus urna. Nunc eu sapien. Donec eget nunc a pede dictum pretium. Proin mauris. Vivamus luctus libero vel nibh.

Fusce tristique risus id wisi. Integer molestie massa id sem. Vestibulum vel dolor. Pellentesque vel urna vel risus ultricies elementum. Quisque sapien urna, blandit nec, iaculis ac, viverra in, odio. In hac habitasse platea dictumst. Morbi neque lacus, convallis vitae, commodo ac, fermentum eu, velit. Sed in orci. In fringilla turpis non arcu. Donec in ante. Phasellus tempor feugiat velit. Aenean varius massa non turpis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Aliquam tortor. Morbi ipsum massa, imperdiet non, consectetur vel, feugiat vel, lorem. Quisque eget lorem nec elit malesuada vestibulum. Quisque sollicitudin ipsum vel sem. Nulla enim. Proin nonummy felis vitae felis. Nullam pellentesque. Duis rutrum feugiat felis. Mauris vel pede sed libero tincidunt mollis. Phasellus sed urna rhoncus diam euismod bibendum. Phasellus sed nisl. Integer condimentum justo id orci iaculis varius. Quisque et lacus. Phasellus elementum, justo at dignissim auctor, wisi odio lobortis arcu, sed sollicitudin felis felis eu neque. Praesent at lacus.

ANEXO C – Fusce facilisis lacinia dui

Vivamus sit amet pede. Duis interdum, nunc eget rutrum dignissim, nisl diam luctus leo, et tincidunt velit nisl id tellus. In lorem tellus, aliquet vitae, porta in, aliquet sed, lectus. Phasellus sodales. Ut varius scelerisque erat. In vel nibh eu eros imperdiet rutrum. Donec ac odio nec neque vulputate suscipit. Nam nec magna. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam porta, odio et sagittis iaculis, wisi neque fringilla sapien, vel commodo lorem lorem id elit. Ut sem lectus, scelerisque eget, placerat et, tincidunt scelerisque, ligula. Pellentesque non orci.

Etiam vel ipsum. Morbi facilisis vestibulum nisl. Praesent cursus laoreet felis. Integer adipiscing pretium orci. Nulla facilisi. Quisque posuere bibendum purus. Nulla quam mauris, cursus eget, convallis ac, molestie non, enim. Aliquam congue. Quisque sagittis nonummy sapien. Proin molestie sem vitae urna. Maecenas lorem. Vivamus viverra consequat enim.