Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Uma introdução sobre novas contribuições em comunidades de software e hardware aberto: Um estudo de caso

Giuliane Eulália Corrêa

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Igor Muzetti Pereira

Março, 2023
João Monlevade–MG

Giuliane Eulália Corrêa

Uma introdução sobre novas contribuições em comunidades de software e hardware aberto: Um estudo de caso

Orientador: Igor Muzetti Pereira

Monografia apresentada ao curso de Engenharia de Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Março de 2023

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Giuliane Eulália Corrêa

Uma introdução sobre novas contribuições em comunidades de software e hardware aberto: Um estudo de caso

Monografia apresentada ao Curso de Engenharia de Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Engenharia de Computação

Aprovada em 3 de Abril de 2023

Membros da banca

MSc. Igor Muzetti Pereira - Orientador - Universidade Federal de Ouro Preto
Dr. Harlei Miguel Arruda Leite - Universidade Federal de Ouro Preto
Fábio Souza - Portal Embarcados

Igor Muzetti Pereira, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 03/04/2023.

Documento assinado eletronicamente por **Igor Muzetti Pereira**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 03/04/2023, às 13:39, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0503207** e o código CRC **DB06C521**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0819 - www.ufop.br

Este trabalho é dedicado à todas as mulheres, principalmente para as que tenham interesse em trabalhar na área de computação, desejo que sempre conquistem seu espaço e que acreditem no seu potencial.

As mulheres não devem ter medo de ser inteligentes.

Agradecimentos

Agradeço ao meu pai, Gilson. Que sempre acreditou em mim e me apoiou em todos os momentos.

"Uma pessoa saber programar é poderosa, uma pessoa que sabe como os negócios funcionam é imparável "

—Zeno Rocha,
in: 14 hábitos de programadores altamente produtivos.

Resumo

Este texto inclui uma visão geral da comunidade Franzininho, uma comunidade brasileira focada em hardware de código aberto, e discute o desenvolvimento de um sistema de irrigação automatizado usando a placa Franzininho, que é uma placa de licença de hardware aberto. O texto discute as lições aprendidas e os desafios enfrentados por uma nova colaboradora da comunidade Franzininho e recomenda maneiras de incentivar mais colaboração. Atualmente no mercado brasileiro de sistemas embarcados e Internet das Coisas (IoT), existem comunidades de computação com a missão de promover a colaboração de projetos de Sistemas IoT que utilizam conceitos de hardware e software livre. Esse estudo busca identificar e discutir os desafios e oportunidades neste mercado e descreve o papel do hardware e software livre que permite a colaboração e inovação na área de IoT. Com o uso de uma placa de desenvolvimento denominada Franzininho, implementamos um sistema de irrigação automatizado para colaborar e interagir com a comunidade, realizamos a montagem do protótipo de hardware com sensores e autuadores elétricos em um software em Circuitpython com as regras de negócio que foram planejadas.

Palavras-chaves: Hardware aberto. Software Livre. Franzininho.

Abstract

This text includes an overview of the Franzininho community, a Brazilian community focused on open source hardware, and discusses the development of an automated irrigation system using the Franzininho license plate, which is an open hardware license plate. The text discusses the lessons learned and challenges faced by a new contributor to the Franzininho community and recommends ways to encourage further collaboration. Currently, in the Brazilian market for embedded systems and the Internet of Things (IoT), there are computing communities with the mission of promoting the collaboration of IoT Systems projects that use hardware and free software concepts. This study seeks to identify and discuss the challenges and opportunities in this market and describes the role of hardware and open source software that enables collaboration and innovation in the field of IoT. With the use of a development board called Franzininho, we implemented an automated irrigation system to collaborate and interact with the community, we carried out the assembly of the hardware prototype with sensors and electrical actuators in a software in Circuitpython with the business rules that were planned .

Key-words: Open hardware. Open software. Franzininho.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

IoT Internet das Coisas

OSH Open Source Hardware

OSS Open Source Software

OSHWA Open Source Hardware Association

GNU GNU General Public License, ou GPL

FSF Fundação para o software livre

FLOSS Free and Open Source Software

IDE Integrated Development Environment

PR Pull Request

USB Universal Serial Bus

EEPROM Electrically-Erasable Programmable Read-Only Memory

GPLv3 GNU General Public License version 3

BNDES Banco Nacional de Desenvolvimento Econômico e Social

Sumário

1 Introdução

Esta pesquisa é sobre um estudo de caso na comunidade Franzininho[1]. A placa Franzininho WiFi foi utilizada para construir um protótipo, ela possui licença de *hardware* aberto e é um projeto desenvolvido pela comunidade.

Essa é uma comunidade ativa de colaboradores nas redes sociais, e ao explorar a comunidade, analisamos os processos e também buscamos estimular a abertura de projetos por meio do entendimento de como funcionam as licenças e publicações. Esse estudo nos possibilitou construir um código de software aberto, utilizando a placa *open hardware* (placa Franzininho WiFi). Foi desenvolvido um sistema de irrigação automatizado com sistema de precisão sem envolvimento humano, utilizando sensores para detectar o teor de umidade do solo através de sinais analógicos e digitais com a placa da comunidade Franzininho.

A comunidade Franzininho tem como objetivo capacitar pessoas para desenvolver projetos eletrônicos. Eles acreditam que projetos de código aberto com ensino aberto ou *open study* (um termo que se refere ao acesso livre e gratuito a material educacional, geralmente disponibilizado na internet). Isso inclui cursos, livros, vídeos e outras formas de conteúdo educacional. O objetivo do estudo aberto é fornecer acesso à educação de qualidade a pessoas que, de outra forma, não teriam acesso a ela. Ele também é conhecido como educação aberta, ensino aberto e aprendizagem aberta, com compartilhamento de conhecimento e experiência geram colaboração capaz de impactar positivamente nossa sociedade.

A fonte de inspiração dessa comunidade é a cultura hacker, com foco em maximizar e aprender novas possibilidades de uso de software e hardware. Isso significa a possibilidade de liberdade de execução, cópia, distribuição, estudo, alteração e aperfeiçoamento do software. Franzininho é uma plataforma de hardware e software de código aberto para dispositivos IoT.

A Franzininho foi desenvolvida durante o Arduino Day em São Paulo (2017). Thalis Antunes e Fábio Souza conceberam o projeto a partir do desafio de realizar atividades simultâneas nos 12 FABLABs livres em SP, em um Arduino Day realizado pela Prefeitura de São Paulo. Ela destina-se a ser utilizado como uma plataforma de exemplos para construção de aplicações IoT. Ela foi projetada para ser fácil de usar e permitir que os usuários criem e testem rapidamente protótipos de IoT. A placa Franzininho é baseada na plataforma *Arduino* e é compatível com uma ampla gama de sensores e atuadores, tornando-o uma escolha popular para desenvolvedores de Internet das Coisas (IoT) no

[1] https://franzininho.com.br/

Brasil e no mundo. No site da comunidade na área de projetos é possível acompanhar a atuação deles para a sociedade.

A comunidade de *hardware* aberto no Brasil ainda é relativamente pequena, mas vem crescendo nos últimos anos. Um dos principais desafios enfrentados pela comunidade é o conhecimento limitado das pessoas em relação a compreensão das licenças de código. Uma das principais lições aprendidas com a comunidade de hardware aberto no Brasil é a importância do envolvimento e colaboração da comunidade.

No geral, a comunidade de design de hardware aberto tem muito potencial de crescimento, mas requer mais suporte e recursos para realizar plenamente seu potencial. Incentivar a colaboração, a educação e o envolvimento da comunidade podem ajudar no enfrentamento dos desafios encontrados pela comunidade e promover o desenvolvimento de projetos de hardware aberto bem-sucedidos.

Nesse estudo também vamos apresentar sobre as licenças, a Licença GNU GPL que representa software livre (ou código aberto), e a Licença *CERN* para hardware livre (ou hardware aberto) que gera proteção sob licença. A GNU [2] GPL garante que o usuário final tenha a liberdade de estudar, compartilhar, copiar e modificar este software. Esse projeto foi lançado em 1983 por *Richard Stallman* com objetivo de criar um sistema operacional completamente livre, incluindo software livre para todos os componentes, incluindo o kernel. O projeto foi fundamental para o surgimento do sistema operacional GNU/Linux e é considerado uma das principais forças por trás do movimento de software livre e de código aberto.

É importante observar que o hardware difere do software, porque os recursos físicos são empregados na produção. A produção de hardware é feita a partir de um design, o design deve fornecer as mesmas liberdades que definem o software livre. O hardware custa dinheiro para ser produzido, então o hardware não será cem por cento gratuito, mas isso não impede que seu design seja gratuito. Em termos éticos, a questão da liberdade supera a questão do preço.

Hardware de código aberto, Open Source Hardware (OSH), refere-se a produtos físicos projetados e distribuídos de forma que os planos e materiais de origem necessários para criá-los estejam disponíveis para qualquer pessoa. Isso inclui não apenas o hardware em si, mas também os arquivos de design, esquemas e outras documentações relacionadas ao produto. O hardware de código aberto permite que os usuários modifiquem, melhorem e distribuam o hardware como acharem melhor, desde que sigam os termos da licença de código aberto sob a qual o hardware é lançado. Isso pode levar à criação de novos produtos e inovações que talvez não fossem possíveis com o hardware tradicional de código fechado. Existem várias práticas recomendadas e comumente seguidas na comunidade de hardware

[2] https://www.gnu.org/

de código aberto. Essas incluem:

- Utilização de licenças de código aberto: isso permite que outros usem, modifiquem e distribuam o hardware, desde que sigam os termos da licença.
- Fornecer documentação detalhada: isso inclui arquivos de design, esquemas e outros materiais que permitem que outras pessoas entendam como o hardware funciona e como construí-lo.
- Utilização de componentes padronizados e bem documentados: isso torna mais fácil para outras pessoas replicarem e modificarem o hardware.
- Incentivar a colaboração e o compartilhamento: o hardware de código aberto conta com uma comunidade de usuários e desenvolvedores que podem contribuir com ideias, melhorias e suporte ao projeto.
- Ser transparente sobre quaisquer riscos ou limitações potenciais: isso permite que outras pessoas tomem decisões informadas sobre a usabilidade do hardware e como utilizá-lo com segurança.

POZZEBOM (2021) explica como os sistemas embarcados são sistemas de computador especializados projetados para executar uma tarefa específica ou um conjunto de tarefas. Eles são frequentemente encontrados em dispositivos como automóveis, eletrodomésticos e equipamentos industriais e geralmente são incorporados ao dispositivo como um único componente ou como parte de um sistema maior. A Internet das Coisas Internet das Coisas (IoT) refere-se à crescente rede de dispositivos conectados equipados com sensores e outras tecnologias que permitem que eles se comuniquem e troquem dados entre si e com sistemas externos. A Oracle (2017) em seu texto comenta que isso inclui uma ampla gama de dispositivos, como termostatos inteligentes, aparelhos inteligentes e equipamentos industriais.

É provável que o mercado de sistemas embarcados e IoT continue crescendo no Brasil e no mundo nos próximos anos, à medida que cada vez mais dispositivos estão conectados à internet e a demanda por automação e tomada de decisão baseada em dados aumenta. De acordo com o site de embarcados[3] (que também é uma plataforma aliada da comunidade Franzininho), o mercado de sistemas embarcados e IoT é uma indústria em rápido crescimento que envolve o uso de dispositivos eletrônicos miniaturizados, geralmente com capacidade de computação e conectividade, para monitorar, controlar e automatizar processos. Esses dispositivos são incorporados em uma variedade de aplicações, como veículos, dispositivos médicos, equipamentos industriais, sistemas de segurança e automação residencial.

[3] https://embarcados.com.br/

A *IoT* é uma das principais tendências do mercado de sistemas embarcados de acordo com a Mordor Intelligence (2022), pois permite que esses dispositivos sejam conectados à internet e troquem dados, tornando possível coletar e analisar grandes quantidades de dados em tempo real. Isso tem o potencial de melhorar significativamente a eficiência, segurança e conveniência em vários setores, como saúde, transporte, produção, agricultura e logística e no Brasil poderá representar receitas de cerca de US$200 bilhões a partir de 2025, segundo estudo do Banco Nacional de Desenvolvimento Econômico e Social (BNDES). Essa tendência é impulsionada pela crescente demanda por dispositivos conectados, automação e análise de dados, bem como pelo aumento da adoção de tecnologias como 5G e inteligência artificial.

Usando componentes padronizados e bem documentados, o uso desses componentes pode facilitar a replicação e modificação do hardware por outras pessoas. Assim como aderir aos princípios de design estabelecidos e às melhores práticas. Sendo assim seguir esses princípios e práticas, como os descritos na Definição de Hardware Aberto da Open Source Hardware Association (OSHWA), pode ajudar a garantir que os projetos sejam bem projetados e confiáveis. Essa é uma organização sem fins lucrativos que tem como objetivo promover e defender a criação, uso e compartilhamento de hardware aberto. Ela é responsável por manter a norma que é uma definição de hardware aberto. Essa norma é uma lista de critérios que devem ser atendidos para que um projeto de hardware possa ser considerado aberto. Esses critérios incluem a disponibilidade dos desenhos e especificações do hardware, a liberdade de estudar, modificar e distribuir o projeto e a obrigação de fornecer informações sobre o uso de componentes patenteados.

A adesão à norma OSHWA é voluntária, mas as empresas, organizações e indivíduos que cumprem esses critérios podem se registrar com ela e usar o selo OSHWA para indicar que seus projetos cumprem os padrões de hardware aberto. Essa norma é importante pois ajuda a garantir que o hardware aberto seja acessível, transparente e colaborativo, tornando-o mais fácil para as pessoas usarem, modificarem e distribuírem. Isso pode impulsionar a inovação e a criatividade, bem como ajudar a reduzir os custos e aumentar a eficiência no desenvolvimento de novos projetos. De forma geral, abaixo vamos abordar a história do projeto, a comunidade do projeto, as aplicações de uso envolvendo hardware e software, os desafios agregados de oportunidades e como esses foram abordados, além das oportunidades para o futuro do projeto.

1.1 O problema de pesquisa

O problema será fazer uma analise de como as comunidades abertas interagem com os membros, anotar as etapas desse estudo de caso e pontua-las, observar a comunidade e poder agregas com essa interação e por fim documentar os resultados alcançados.

Para poder contribuir e ingressar na comunidade como uma membra, foi iniciado o desenvolvimento de um sistema de irrigação automático com a placa fornecida pela comunidade, identificamos as dificuldades de desenvolvimento, a interação com os membros, identificamos como realizar e documentar o *pull request*, para poder publicar pelas regras da comunidade seguindo as normas *open source* e por fim discutir todas as lições aprendidas e ideias de como colaborar. Sendo assim, podendo relatar a experiência de como ingressar e contribuir em uma comunidade de hardware aberto brasileira.

1.2 Objetivos

Explorar a comunidade de hardware aberto no Brasil, identificando desafios e discutindo lições aprendidas, analisando os processos e incentivando a abertura de projetos entendendo como funcionam as licenças e publicações, para isso utilizaremos como estudo de caso a comunidade do projeto de hardware aberto Franzininho. Poder analisa-las para poder descobrir como pode ser benéfico para a ciência e agregar novas descobertas, junto com analise dos recursos econômicos e sociais, e os resultados que essas comunidades geram no mercado. Para superar os desafios, foi importante ter acesso aos recursos e estudos sobre licenças de código aberto e como usá-las e atribuí-las adequadamente em projetos de hardware. Além disso, promover a colaboração e o compartilhamento dentro da comunidade pode ajudar a reunir recursos e conhecimento, levando a projetos de hardware aberto mais bem-sucedidos.

1.3 Organização do trabalho

Apresentaremos a revisão da literatura apontando pesquisas e análises de fontes relevantes sobre o assunto, incluindo artigos e livros especializados. No capitulo 3, temos a descrição da metodológica utilizada para coletar e analisar dados, incluindo as técnicas de coleta de dados e análises. Em seguida, o desenvolvimento prático é mostrado e como foi aplicado. No capitulo 4 em resultados, apresentamos algumas respostas da pesquisa e alguns recursos que foram implementados. Por fim no capitulo 5 a conclusão com uma discussão, síntese dos resultados e comparação com a revisão da literatura de acordo com a interpretação dos resultados e das conclusões, incluindo a resposta à hipótese inicial. Em referências, no capitulo 6 estão listadas todas as fontes utilizadas e referenciadas no texto, no capitulo 7 os anexos com materiais adicionais, como fotos do projeto que complementam o trabalho.

2 Revisão bibliográfica

The Cathedral and the Bazar de Raymond (1999) é um conhecido ensaio do desenvolvedor de *software* e defensor do código aberto, publicado pela primeira vez em 1997. No ensaio, *Raymond* compara o modelo tradicional de ”catedral” de desenvolvimento de software, no qual um pequeno grupo de desenvolvedores trabalha em um projeto de forma isolada, para o modelo ”bazar” no qual uma grande comunidade de voluntários trabalha em um projeto de forma colaborativa. *Raymond* argumenta que o modelo de bazar é mais eficaz e eficiente do que o modelo de catedral, porque permite um desenvolvimento mais rápido e uma inovação mais rápida. Ele cita o sucesso do *Kernel Linux* como exemplo do poder do modelo bazar.

Em geral, a padronização de práticas no desenvolvimento pode ajudar a garantir que os projetos sejam de alta qualidade, seguros e fáceis de usar e modificar e compatíveis com outros hardwares e softwares. O mercado do *Kernel Linux* também entra como exemplo a ser mencionado neste artigo por ser um *Kernel* de código aberto que é desenvolvido e mantido por uma grande comunidade de voluntários, com coordenação da *Linux Foundation*. O *kernel* é o componente central de um sistema operacional e é responsável por gerenciar os recursos de hardware do computador e fornecer serviços para as outras partes do sistema operacional. O desenvolvimento do *Kernel Linux* segue um modelo de "bazar", no qual qualquer pessoa pode contribuir com código para o projeto, e as contribuições são revisadas e integradas por uma equipe de mantenedores. Este modelo tem tido muito sucesso em permitir que o *kernel* evolua e se adapte rapidamente a novas tecnologias e requisitos. Com o tempo, o processo de desenvolvimento do *kernel Linux* tornou-se mais formalizado, com um conjunto definido de regras e procedimentos para contribuir com código e resolver disputas. O número de contribuidores e o tamanho do *kernel* também aumentaram drasticamente, levando a alguns desafios no gerenciamento do projeto e garantindo que o *kernel* permanecesse estável e de alta qualidade.

No artigo de Morreale1 (2017) *“Construir comunidade ao redor de uma plataforma aberta”*, foi possível analisarmos como a construção de criadores em torno de uma plataforma de hardware aberta que envolve várias etapas, sendo uma delas certificarmos de que seja de código aberto e facilmente modificável por outros, e incentivar as pessoas a compartilharem seus projetos e modificações na plataforma de hardware com outras pessoas. Os processos de design técnico e construção da comunidade são detalhados, culminando em uma campanha de *crowdfunding* bem-sucedida A partir das observações e entrevistas, refletimos sobre a relação entre a plataforma e a comunidade e oferecemos sugestões para pesquisadores e praticantes interessados em estabelecer suas próprias comunidades de makers.

Como vimos no ensaio do livro do Raymond, Melissa Wen (2021) também pontua sobre ambientes de desenvolvimento aberto em sua dissertação de mestrado, onde ela aborda um estudo aprofundado sobre o modelo atual de desenvolvimento do *Kernel Linux*, especificamente, o que acontece quando o bazar (comunidade de desenvolvedores) cresce. A dissertação estuda as práticas atuais de desenvolvimento do *kernel Linux* como parte do fenômeno de Software Livre Free and Open Source Software (FLOSS). O estudo se concentra em cobrir a perspectiva tanto dos acadêmicos quanto dos profissionais envolvidos no desenvolvimento, incluindo a revisão de literatura multi vocal e a observação participante. O objetivo é mapear as características atuais da comunidade de desenvolvimento do *kernel* Linux e sintetizar estratégias de pesquisa para futuros estudos de ecossistemas FLOSS. Como resultado, a dissertação apresenta o estado da arte e o estado da prática do modelo de desenvolvimento contemporâneo do *kernel Linux* e oferece uma combinação de métodos de pesquisa para futuras investigações.

Para entendermos mais como as comunidades funcionam buscamos alguns artigos que falavam sobre o comportamento dos membros. No artigo de Steinmacher (2013) em *"Why do newcomers abandon open source software projects?"*- Igor Steinmacher aborda o problema da rotatividade dos novatos em projetos de software de código aberto. Ele examina as principais razões para a saída de novos membros e propõe algumas soluções que podem ajudar a melhorar a experiência dos novos usuários, como o uso de sistemas de *feedback*, treinamento adequado e uma cultura colaborativa. Em resultados eles conseguiram montar um modelo conceitual composto por 38 barreiras, agrupadas em sete diferentes categorias. Essas barreiras podem motivar novos estudos e o desenvolvimento de ferramentas apropriadas para melhor apoiar a integração de novos colaboradores. Seguindo essa linha, podemos analisar o artigo Fronchetti (2013) em *"What Attracts Newcomers to Onboard on OSS Projects?"* que examina o que atrai novos usuários para se juntar a projetos open source Open Source Software (OSS). Ele descobriu que a popularidade é um fator importante, pois pode ajudar a atrair novos usuários para contribuir com o projeto. O estudo também mostrou que os recursos disponíveis em OSS e a documentação são essenciais para garantir o sucesso dos projetos. Foi descoberto que a popularidade do projeto (em termos de estrelas), o tempo para revisar as solicitações *pull*, a idade do projeto e as linguagens de programação são os fatores que melhor explicam os padrões de crescimento dos recém-chegados.

Essas comunidades também apoiam a diversidade, e em “Uma investigação empírica sobre os desafios enfrentados pelas mulheres na indústria de software: um estudo de caso”, de Trinkenreich (2022). Examina-se os desafios enfrentados pelas mulheres no setor de software, com base em entrevistas e observações realizadas em uma grande empresa de tecnologia. Os resultados mostraram que as mulheres enfrentam muitos desafios relacionados a questões de gênero, como falta de reconhecimento por suas habilidades e contribuições, assédio sexual e discriminação. Além disso, o estudo identificou barreiras institucionais que dificultam

o progresso das mulheres na indústria, como a falta de políticas específicas para apoiar mulheres no mercado de trabalho. Finalmente, o estudo analisou várias recomendações para promover a igualdade de gênero nos ambientes de trabalho da indústria de software. As mulheres enfrentam desafios significativos na indústria de software, pois ainda há desigualdades de gênero e falta diversidade nas equipes de criação. Isso também afeta a inclusão das mulheres nos níveis mais altos da liderança, onde elas ainda estão sub-representadas em comparação aos homens. No entanto, as mulheres ainda se deparam com barreiras que limitam suas possibilidades e crescimento profissional, especialmente na indústria de software. Estes obstáculos incluem discriminação salarial, falta de acesso a treinamento e recursos, e falta de representação no nível das decisões. Além disso, muitas vezes as mulheres precisam lutar contra estereótipos negativos dentro dos ambientes industriais.

Por essas razões, é fundamental incentivar a diversidade nos ambientes de trabalho para promover o crescimento e o sucesso em grande escala para todos os envolvidos. Embora o setor industrial esteja se movendo para tornar as equipes mais diversificadas e abertas às contribuições de todos, ainda existem grandes desafios que precisam ser superados para que as mulheres tenham igualdade de oportunidades. A diversidade é importante não apenas para promover a igualdade, mas também para melhorar a produtividade, a inovação e o sucesso empresarial. É essencial que as empresas reconheçam o valor da diversidade e estabeleçam políticas que promovam igualdade de oportunidades para todos. O Projeto Emílias Armação em Bits [1], é uma iniciativa que promove a diversidade de gênero na computação. A organização visa destacar a contribuição das mulheres na história e na atualidade da computação ao realizar entrevistas com mulheres da área. O projeto visa divulgar o trabalho dessas mulheres para inspirar a próxima geração de profissionais de computação e incentivar a diversidade no setor.

Vamos citar em alguns momentos desse estudo as licenças utilizadas em comunidade abertas, então é importante relacionar esse estudo a Licença Pública Geral GNU General Public License, ou GPL (GNU) *(ou GPL)* é uma licença de software livre criada pela Fundação para o Software Livre (FSF) em 1989. Na plataforma oficial da pagina da GNU foi possível encontrar, que ela é utilizada para garantir que o software liberado sob essa licença seja sempre livre e aberto. A GPL é uma licença copyleft, o que significa que todas as versões modificadas e distribuídas do software devem ser licenciadas sob os mesmo termos da GPL. Isso garante que qualquer alteração feita no software seja também livre e aberta. Além disso, a GPL permite a combinação de software licenciado sob a GPL com outros programas, desde que esses programas também sejam licenciados sob termos compatíveis com a GPL. A GPL é uma das licenças de software livre mais populares e amplamente utilizadas no mundo, sendo utilizada em muitos projetos de software, incluindo

[1] https://www.youtube.com/@EmiliasArmacaoemBits

o sistema operacional Linux, o GIMP e o LibreOffice. Ela é uma das licenças recomendadas pela *Free Software Foundation* Fundação para o software livre (FSF) e é considerada uma das licenças de software livre mais respeitadas e confiáveis.

A GNU General Public License version 3 (GPLv3) é a terceira versão da Licença Pública Geral GNU da Fundação para o Software Livre (FSF). Ela foi lançada em 2007 como uma atualização da versão anterior, a GPLv2, e é utilizada para garantir que o software liberado sob essa licença seja sempre livre e aberto, e que as pessoas tenham permissão para usar, estudar, modificar e distribuir esse software. A em Smith (2022) sobre a GPLv3, também aborda sobre ela ser uma licença copyleft, o que significa que todas as versões modificadas e distribuídas do software devem ser licenciadas sob os mesmos termos da GPLv3. Isso garante que qualquer alteração feita no software seja também livre e aberta. A GPLv3 é amplamente utilizada em muitos projetos de software, incluindo o sistema operacional Linux, o GIMP e o LibreOffice. Ela é uma das licenças recomendadas pela *Free Software Foundation* Fundação para o software livre (FSF) e é considerada uma das licenças de software livre mais respeitadas e confiáveis.

Conforme foi publicado por Souza (2019) em *Open source hardware: Conheça a definição e as boas práticas* o termo *"Open Source Hardware (OSHW)"* é utilizado para artefatos tangíveis — máquinas, dispositivos ou outros objetos físicos — cujo projeto foi disponibilizado ao público de modo que qualquer um possa construir, modificar, distribuir e utilizar estes artefatos. É a intenção desta definição auxiliar no desenvolvimento de guias gerais para o desenvolvimento e validação de licenças para *Open Source Hardware*.

A comunidade também disponibilizam alguns artigos que nos ajudaram a entender as definições e algumas boas práticas, no Brasil ainda são poucas as iniciativas de projetos *open hardware* e a comunidade brasileira ainda possui dúvidas quanto a publicação, contribuição e uso de projetos abertos.

3 Metodologia

O primeiro contato com a comunidade foi reunir com o fundador da Franzininho, discutir a potência da comunidade e a forma que ela atua nas redes sociais. Fábio, um dos diretores e fundadores, nos enviou alguns links informativos, como por exemplo, para adquirir a placa, a documentação para introduzir os primeiros passos, as redes sociais que eles atuam como *Telegram, Discord* e blog com alguns exemplos de *hardware* livre.

Nessa conversa também discutimos as pesquisas sobre o mercado brasileiro de trabalhos embarcados, a comunidade tem alguns dados de 2014 a 2021 sobre sistemas embarcados e IoT, esses dados coletados falam sobre o processo de desenvolvimento, tecnologia e fabricantes aplicados em projetos embarcados através de profissionais que atuam na área. Eles também falam que estão interessados em saber quais tipos de aplicação estão sendo desenvolvidas, quais ferramentas que são utilizadas e quais tecnologias de internet das coisas estão sendo mais empregadas.

Essa pesquisa realizou perguntas aos desenvolvedores convidados por *e-mail* e foi enviada para a base de cadastro de dados da comunidade de embarcados e alguns parceiros. Esses questionários respondem dentre algumas questões quais são os perfis profissionais que atuam diretamente com desenvolvimento de sistemas embarcados. Essa pesquisa revela números importantes, como o domínio na área ainda ser de homens, e mais da metade serem desenvolvedores/engenheiros que predominam em desenvolver nas áreas que atuaram e frequentaram durante a universidade. A comunidade tem alguns artigos que nos ajudaram a entender as definições e algumas boas práticas, no Brasil ainda são poucas as iniciativas de projetos *open hardware* e a comunidade brasileira ainda possui dúvidas quanto a publicação, contribuição e uso de projetos abertos.

Nesse projeto vamos mostrar na prática como aproveitar um hardware livre e no fim fazer uma colaboração completa me tornando uma colaboradora da comunidade. Com a placa em mãos começamos a construir um sistema funcional amigável para iniciantes, e manter as plantas vivas usando sensores, microcontroladores e um software livre.

A comunidade de hardware aberto no Brasil é composta por indivíduos e organizações de uma ampla variedade de origens e setores. A transferência de conhecimento nesse ambiente possui um vasto potencial de disseminação de conhecimento e experiência em design e desenvolvimento de hardware. A colaboração com esta comunidade pode oferecer oportunidades para aprender com os outros e melhorar as próprias habilidades e conhecimentos. Essas comunidades são globais, o que significa que colaborar com a comunidade de hardware aberto no Brasil pode fornecer acesso a uma comunidade global de especialistas e recursos que podem ser aproveitados para desenvolvimento pessoal

ou profissional. Ao apoiar e colaborar com a comunidade de hardware aberto no Brasil, ajuda-se a promover o desenvolvimento da indústria local e criar oportunidades para indivíduos e organizações no país.

O primeiro contato com a placa Franzininho WiFi foi realizando alguns testes, com *framework* diferentes para buscar entender qual linguagem seria melhor para esse trabalho, realizamos testes na linguagem C e *Python* antes de escolhermos o *CircuitPhython*, também foi realizada as instalações que podem serem vistas em no capitulo 3.8 antes de construir o sistema como um todo e decidir em qual linguagem iríamos programar.

A comunidade ofereceu um suporte inicial com testes em algumas linguagens na documentação do hardware, realizam vídeos no *youtube*, além de ficarem à disposição para auxiliar em dúvidas na instalação através dos canais de interação como *Telegram*. Foi preciso fazer algumas adaptações/instalação na placa Franzininho WiFi para atender as necessidades do nosso projeto, essas adaptações são citadas em no capitulo 3.8, onde citamos os primeiros passos com a placa.

No caso desse sistema, empregamos a distribuição Ubuntu GNU/Linux para configurar a placa Franzininho WiFi. Ubuntu[1] é um sistema operacional ou sistema operativo de código aberto, construído a partir do núcleo Linux. Para usar a linguagem na placa basta fazer *download* da biblioteca *CircuitPython* com auxílio de um Universal Serial Bus (USB) e instalar na placa o programa com a biblioteca.

Para contextualizar o cenário, é de importância ter um contato inicial com alguns conceitos que estão apontados abaixo na seção 3.1 a 3.5, esses conceitos são *frameworks* suportados pela placa que podem ser manuseados para realizar aplicações e que serão citados ao decorrer desse estudo. E nos capítulos 3.8 e 3.9 vai ser abordado como foi montado o experimento

3.1 ESP32-S2

O ESP32-S2 é um *WiFi SoC* (System on Chip). Ele foi projetado para uso em uma variedade de aplicativos *IoT* e pode ser programado usando o Arduino como Integrated Development Environment (IDE) ou outras ferramentas de desenvolvimento de software. Após o *upload* do código, você pode usar o monitor serial para visualizar a saída do seu código e enviar dados para a placa ESP32-S2.

3.2 CircuitPython

CircuitPython é uma linguagem de programação de código aberto baseada em *Python* e foi projetada para ser fácil de usar e fácil de aprender. É frequentemente usado

[1] https://ubuntu.com/download

para programar microcontroladores e outros pequenos computadores de placa única, como o *Adafruit Circuit Playground Express* e o *Raspberry Pi*. A documentação do *CircuitPython* está disponível no seu site[2] oficial e no *GitHub*. A documentação inclui informações sobre como instalar e configurar o *CircuitPython*, bem como tutoriais e exemplos, que mostram como usar essa linguagem para controlar hardware e construir projetos. A documentação é organizada em várias seções, incluindo um guia de introdução, um manual de referência e uma coleção de exemplos e tutoriais.

Há também um *CircuitPython Library Bundle* voltado para a comunidade, que inclui um grande número de bibliotecas e módulos que podem ser usados para estender os recursos do *CircuitPython*. No geral, a documentação dessa ferramenta é extensa e abrangente, ela foi projetada para ajudar usuários de todos os níveis de habilidade a aprender como usar a linguagem e criar projetos com ele.

3.3 Mu-Editor

Mu[3] é um editor de código simples, projetado para programadores iniciantes e inclui um interpretador e depurador *Python* integrado. O Mu-editor está disponível para *Windows, Mac, Linux* e *Raspberry Pi* e foi projetado para ser fácil de usar e fácil de aprender.

A documentação do Mu-edito está disponível no site oficial da Mu-editor e no *GitHub*. A documentação inclui informações sobre como instalar e configurar o Mu, bem como tutoriais e exemplos que mostram como usar o Mu para escrever e executar o código *Python*. A documentação é organizada em várias seções, incluindo um guia de introdução, um manual de referência e uma coleção de exemplos e tutoriais. Há também um fórum da comunidade onde os usuários podem fazer perguntas e obter ajuda para usar o Mu. No geral, a documentação do Mu-editor é extensa e abrangente e foi projetada para ajudar os programadores iniciantes a aprender como usar o Mu-editor e começar a programar em *Python*.

3.4 Espressif

A Espressif[4] Systems é uma empresa chinesa especializada no desenvolvimento de tecnologias de comunicação sem fio e *IoT*. Eles oferecem uma gama de produtos, incluindo microcontroladores, módulos *Wi-Fi* e módulos *Bluetooth*, bem como ferramentas de software e kits de desenvolvimento para ajudar os usuários a projetar e construir aplicativos *IoT*. A Espressif fornece documentação para seus produtos em seu site, incluindo folhas de dados,

[2] https://circuitpython.org/
[3] https://codewith.mu/
[4] https://www.espressif.com/

guias do usuário, notas de aplicação e referências de API. Esta documentação destina-se a ajudar os usuários a entender como usar os produtos da Espressif e integrá-los em seus projetos. A documentação é organizada por produto e normalmente está disponível em inglês e chinês. Parte da documentação também está disponível em outros idiomas, como japonês e coreano.

3.5 Internet das coisas

A Internet das Coisas (IoT) refere-se à conexão de dispositivos (que não sejam dispositivos típicos, como computadores e *smartphones*) à Internet e entre si, permitindo que enviem e recebam dados. Esses dispositivos, geralmente chamados de "dispositivos inteligentes", podem incluir uma ampla variedade de itens, como termostatos, alto-falantes, luzes, eletrodomésticos e até roupas. O objetivo da IoT é permitir que esses dispositivos funcionem juntos e tornar a vida das pessoas mais fácil e eficiente. “Em internet das coisas: da teoria à prática”, estudo realizado por alunos da UFMG diz que:

> “internet das coisas, em poucas palavras, nada mais é que uma extensão da Internet atual, que proporciona aos objetos do dia-a-dia (quaisquer que sejam), mas com capacidade computacional e de comunicação, se conectarem à Internet” - Departamento de Ciência da computação, UFMG, Santos (2021).

Há uma variedade de estruturas que são usadas no desenvolvimento e implantação de sistemas IoT. Essas estruturas fornecem um conjunto de diretrizes e ferramentas para criar soluções que podem ajudar a garantir que sejam confiáveis, seguras e escaláveis. É importante determinarmos a estrutura certa para seu caso de uso específico, pois diferentes estruturas são otimizadas para diferentes tipos de aplicativos e redes de IoT.

3.6 Desenvolvimento

Interagindo com a comunidade e buscando projetos exemplos já produzidos pela comunidade, optamos em abordar a construção de um sistema de irrigação automática. Um sistema de irrigação automática é um dispositivo que é usado para regar plantas ou gramados automaticamente, sem a necessidade de intervenção manual. Um sistema de irrigação automática IoT é um tipo de sistema de irrigação automática que está conectado à internet e pode ser controlado e monitorado remotamente usando um smartfone ou outro dispositivo. Existem vários benefícios em usar um sistema de irrigação automática IoT:

- Eficiência: configurado para regar suas plantas somente quando elas precisam, o que pode ajudar a reduzir o uso de água e economizar dinheiro em sua conta de água.

- Monitoramento: pode ser monitorado, para que você possa acompanhar a quantidade de água que suas plantas estão recebendo e fazer os ajustes necessários.
- Controle: você possa ligá-lo ou desligá-lo conforme necessário ou fazer alterações no cronograma de irrigação na placa.

Além dessa motivação, foi possível ver que faltava exemplos completos de sistemas utilizando a placa na pagina da franzininho. Então por fim poderia ser uma possibilidade de colaborar nesse estudo de caso.

Com a placa da Franzininho *WiFi*, começamos a implementar esse projeto, esse artigo também irá demonstrar foi montado o protótipo. Para isso, será necessário uma placa microcontroladora tipo Arduino, com um módulo *WiFi*, como a placa Franzininho *WiFi*, e alguns materiais que vamos listar na tabela 1 abaixo :

Tabela 1 – Níveis de investigação.

Materiais utilizados	**Especificações**
Placa Franzininho WiFi	Hardware com ESP32-S2
Sensor Higrômetro	Sensor de umidade de solo
Protoboard	830 furos, simples
Relé	5V
Mini Bomba de água	5V
Fios jumper	Macho/Macho e Macho/Fêmea

Fonte: Produzido pela autora.

Nota: Os dados foram baseados no protótipo realizado por esse estudo de caso

Aqui vamos falar de alguns passos básicos que seguimos para criar um protótipo. Foi determinado o objetivo do sistema de irrigação e as plantas ou áreas que serão regadas. Isso nos ajudou a selecionar os componentes apropriados a projetar o sistema.

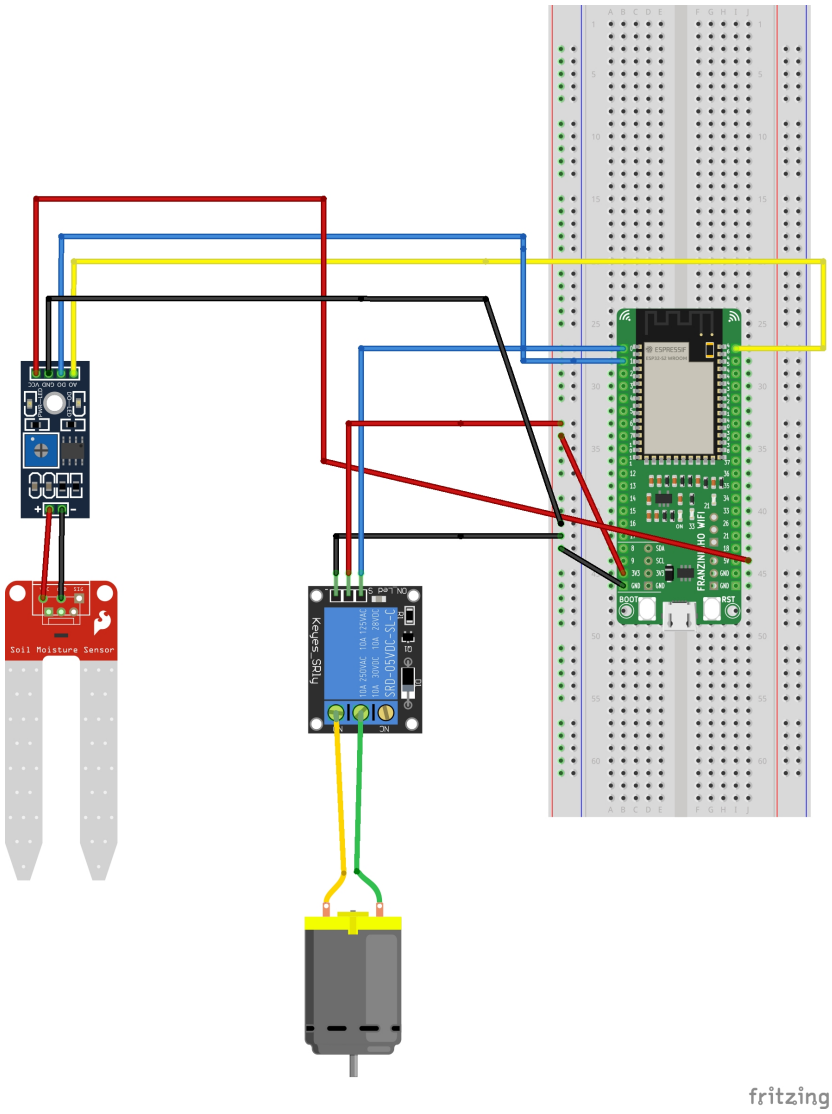
Após reunir os materiais e ferramentas necessários, foi necessária uma placa microcontroladora Franzininho *WiFi*, sensores para medir a umidade e temperatura do solo, uma bomba ou sistema de aspersão para entregar a água e uma fonte de alimentação. Também foram necessários fios, conectores, e uma *breadboard* de prototipagem para a construção do protótipo (materiais listados na tabela 1).

O circuito para o sistema de irrigação foi projetado no Fritzing[5], como pode ser observado na Figura 2, para construir um protótipo do circuito que controlará o sistema de irrigação. Foi necessário escrever um código para o sistema de irrigação. Usamos a linguagem suportada pela placa para escrever o código que vai rodar na placa microcontroladora e controlar o sistema de irrigação.

[5] https://fritzing.org/

Após o desenvolvimento o protótipo do sistema de irrigação foi testado, para garantir que estava funcionando corretamente. E por fim foi, realizado os ajustes necessários no circuito e código. Depois de ter um protótipo funcional, foi possível refinar e melhorar o sistema com base em suas necessidades e preferências. É possível ver a imagem da simulação feita pelo software *Fritzing* na Figura 1.

Figura 1 – Prototipo do sistema de irrigação



3.7 Implementação

A placa Franzininho *WiFi* suporta USB nativo, na maioria das placas permitindo edição de arquivos e ferramentas especiais. Aplicamos a documentação que está no site da Franzininho, a placa foi adicionada no site oficial do CircuitPython[6] e todas as *releases*

[6] https://circuitpython.org/downloads?q=franzininho

da linguagem para compilar nela. Dentro da plataforma da linguagem é possível baixar o arquivo binário para cada tipo de placa.

Essa linguagem é baseada em *MicroPython*, unificadas do *Python*, as bibliotecas também funcionam em computadores de placa única com *Python* e regulam por meio da biblioteca *Adafruit Blinka*. A instalação possibilita o desenvolvimento com o ESP32-S2 usando a linguagem para configurar a placa. Gravamos o *firmware* do *CircuitPython* na Franzininho *WiFi* e preparamos o ambiente para usar a placa.

Para isso bastou entrar no site e fazer download do binário, no nosso caso, foi da *wroom*. Após fazer o download, a placa foi conectada ao computador, e foi necessário pressionar o botão na placa de *bot* e *reset* juntos, em seguida soltar o *reset* e por último o *bot*. Dessa forma a placa entra em modo *DFU*, a partir dessa configuração inicial, foi possível realizar a implementação no dispositivo.

O armazenamento de dados também foi algo a ser analisado, pois o registrador de dados pode precisar armazenar dados por longos períodos de tempo, potencialmente por meses ou anos. Isso requer o uso de mídia de armazenamento não volátil e de baixa potência, como memória *flash* ou Electrically-Erasable Programmable Read-Only Memory (EEPROM).

A placa possui memória flash dentro da ESP32-S2 e ela suporta apenas um arquivo com linhas de instruções por vez. A capacidade de memória flash dentro da ESP32-S2 varia de acordo com o fabricante e o modelo específico. Em geral, a capacidade de memória flash da ESP32-S2 é de aproximadamente 4 MB a 16 MB. No nosso caso a memória é de 4MB. Alguns fabricantes podem oferecer opções de capacidade de memória flash menor ou maior. A memória flash é usada para armazenar o código do programa e os dados do usuário, ela é necessária para a operação do dispositivo e para garantir a persistência de dados.

A transmissão de dados acontece quando o registrador de dados precisa transmitir dados para um local remoto, ele precisará ter um sistema de comunicação sem fio de baixa potência, como *Bluetooth Low Energy* ou uma rede WiFi de baixa potência. Com isso o projeto físico no registrador de dados foi ser projetado para suportar as condições ambientais em que será usado, como temperaturas extremas e exposição à luz solar. Ele também é robusto e durável para garantir que possa suportar o desgaste do uso a longo prazo. No geral, projetar um registrador de dados de potência ultrabaixa para sistemas autônomos requer uma consideração cuidadosa do consumo de energia, armazenamento e transmissão de dados e design físico.

Para instalar o *CircuitPython* no microcontrolador, você pode usar a ferramenta "dfu-util" no Linux ou o "dfu-util" no macOS ou Windows. Depois de instalar o *CircuitPython*, você pode acessar o REPL (Read-Eval-Print Loop) para interagir com o

microcontrolador e começar a escrever códigos em *Python*. Dentro do ambiente, vem um arquivo *.py*, onde basta salvar os códigos nesse arquivo, e a placa os executa automaticamente na inicialização. Além disso, você pode empenhar bibliotecas adicionais para acessar recursos do microcontrolador, como entradas e saídas digitais e analógicas, sensores e atuadores. Depois disso, é possível aplicar a linguagem *CircuitPython* para desenvolver projetos com a placa Franzininho *WiFi*.

3.8 Primeiros passos com a placa

No site da comunidade temos as orientações de como seguir os primeiros passos com a placa, abaixo iremos apontar esses processos. Para gravar o *firmware* na placa, utilizamos a ferramenta esptool.py, essa ferramenta é usada para gravar o *firmware* no microcontrolador ESP32-S2 na placa Franzininho *WiFi*. para instalá-la, usamos o o gerenciador de pacotes *pip*, que vem instalado por padrão com o *Python*:

```
$pip install esptool
```

Depois de instalar a *esptools*, foi possível gravar o *firmware* na placa. Primeiro, conectamos a placa ao nosso computador usando um cabo USB. Em seguida, executamos o seguinte comando:

```
$esptool.py --chip esp32s2 --port /dev/ttyUSB0 write_flash
-z 0x1000 firmware.bin
```

Com esse comando gravamos o *firmware* na placa e reiniciá-la automaticamente. Depois de gravar o *firmware*,foi possível começar a usar a placa Franzininho *WiFi* com *CircuitPython*. Para isso, bastou conectar a placa ao computador usando um cabo USB e acessar a unidade de disco removível que aparecer.

Nela encontramos uma pasta chamada *"lib"* e um arquivo chamado *"code.py"*. Nessa pasta escrevemos o nosso código *Python* no arquivo *code.py* e ele foi executado assim que a placa foi reiniciada.

Para instalar bibliotecas adicionais usando o gerenciador de pacotes *pip*. Bastou conectar a placa ao seu computador usando um cabo USB e executar o seguinte comando:

```
$ pip install <biblioteca>
```

Substituímos a "biblioteca"pelo nome da biblioteca que você deseja instalar. Essa biblioteca ficará disponível para uso no seu código assim que a placa for reiniciada.

3.9 Construindo o sistema de irrigação

Na primeira parte do código temos a importação das bibliotecas a serem usadas na aplicação. Note foi importado o módulo adafruit-dht, instalado na pasta lib da placa:

```
1 import board
2 import time
3 from digitalio import DigitalInOut, Direction, Pull
4 from analogio import AnalogIn    #to import the analog input modul
```

O*digitalio* é um módulo que contém classes para fornecer acesso a IO digital básica, o *touchio* é um módulo que contém classes para fornecer acesso ao *touch IO* normalmente acelerado por hardware no microcontrolador integrado. Todas as classes mudam o estado do hardware e devem ser reinicializadas quando não forem mais necessárias se o programa continuar após o uso.

```
5 relay = DigitalInOut(board.IO0)
6 relay.switch_to_output()
```

O relé foi configurado para usar o portão *Always Close*, então é preciso definir como verdadeiro para que o relé fique inativo. A variável *relay* recebe o *digitalio*. que é um módulo que contém classes para fornecer acesso a IO digital básica e aponta para a porta O0. O sensor de umidade é armazenado na variável *touch*, o *touchio* é um módulo que contém classes para fornecer acesso ao *touch IO* para o hardware no microcontrolador integrado e aponta para entrada O1.

```
7 humid_analog = AnalogIn(board.IO1)
8 humid_digital = DigitalInOut(board.IO4)
9 humid_digital.direction = Direction.INPUT
```

O relé é iniciado como *True*, ele vem como falso de fábrica então é necessário iniciá-lo com valor verdadeiro. O período de tempo para verificar o sistema como espera e o tempo para regar foram definidos como 1s para realizar testes dos valores continuamente de forma mais rápida e termos o retorno imediato, futuramente pode ser alterado para um valor mais alto.

O sensor de umidade do solo (Higrômetro) é um módulo eletrônico desenvolvido com a finalidade de medir variações de umidade do solo. Caso o solo esteja seco o sensor mantém a saída digital em nível alto e quando o solo estiver úmido a saída se mantém em nível baixo.

```
10 relay.value = True
11 wait_time = 1
12 watering_time = 1
13 dry_value = 51130
```

O valor da terra seca será colocado no *dry-value* captado pelo hidrômetro. Para medir a variação da umidade no solo, foi necessário o uso do pino analógico disponível no sensor em conjunto com um microcontrolador que possua conversor analógico digital e girar para celebrá-lo para sinal analógico até que os dois *leds* sejam acesos, isso é feito para que ele reconhecesse a terra quando estivesse úmida e variasse os valores.

Dentro do *while*, na linha 15 do código, está a execução do sistema, recebendo a temperatura e umidade do higrômetro, são armazenados os valores captados e na comparação do *if* analisamos o valor da terra seca e o valor captado e caso esteja abaixo o relé recebe falso e ele envia sinal para a bomba de água pelo tempo que determinamos e depois volta a 'dormir', caso contrário no momento em que for comparado os valores a terra ainda estiver úmida ele entra no *else* e apenas continua dormindo e o relé continua como *true*.

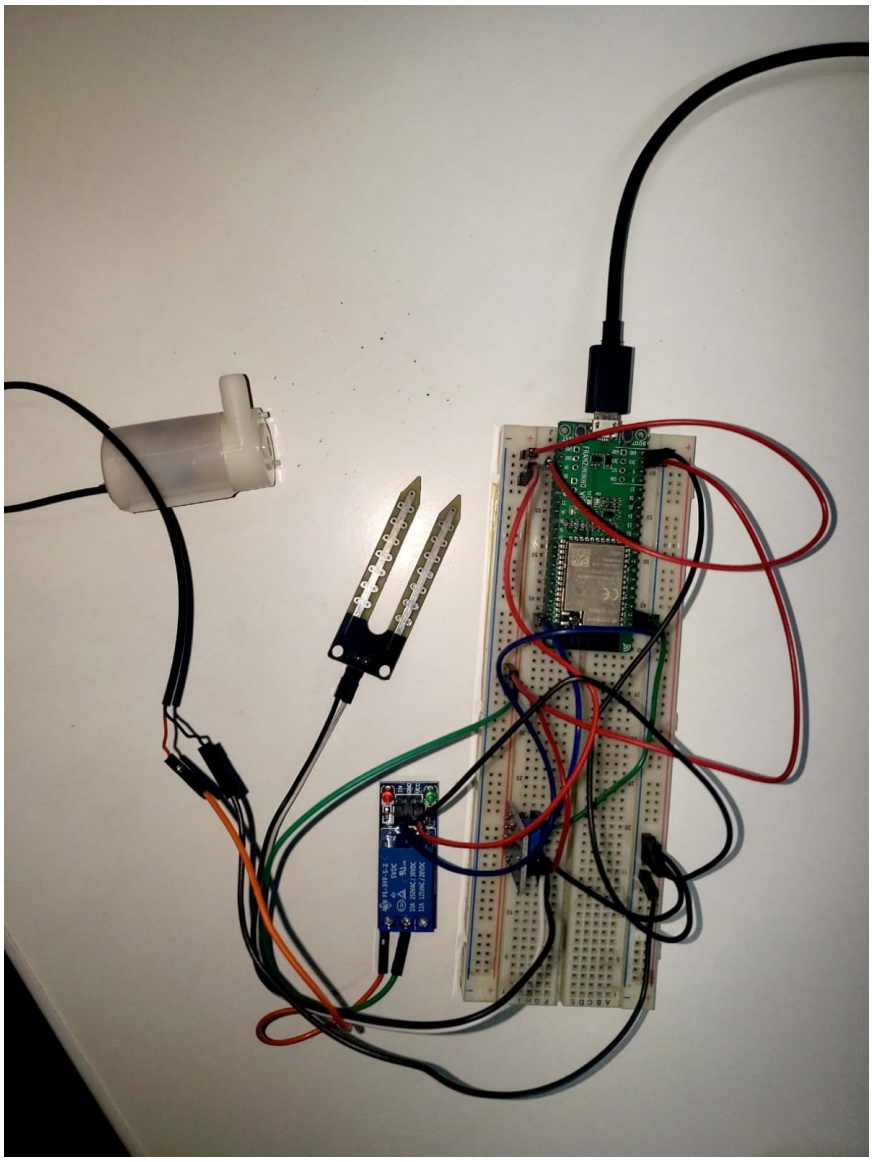
```
14 while True:
15    try:
16        print("humid (Digital value):", humid_digital.value)
17        print("humid (Analogic value):", humid_analog.value)
18
19        time.sleep(1);
20
21        if humid_analog.value > dry_value :
22            print("Starting watering...")
23
24            relay.value = False
25
26            time.sleep(watering_time)
27            print("Finishing watering.")
29
30        else:
31            relay.value = True
32            time.sleep(wait_time)
33
34    except RuntimeError as e:
35        print("Read failure")
```

```
36
37 time.sleep(1)
```

O código completo pode ser conferido no *github*:<https://github.com/GiulianeEC/SystemOfIrrigation_Franzininho>

O video do sistema funcionando no Youtube:<https://www.youtube.com/shorts/kXlZjeyKTdk>

Figura 2 – Imagem real do protótipo



3.10 Pull Request

Como parte de pesquisa, foi proposto realizar um *pull request* na página oficial realizando uma publicação sobre o protótipo do projeto de irrigação realizado na placa, podendo passar pela experiencia de contribuir oficialmente e relatar todo o processo.

Um Pull Request (PR) é um recurso do *Git* e do *GitHub* que permite aos desenvolvedores propor alterações em um projeto e colaborar com outras pessoas no desenvolvimento do código. Quando um desenvolvedor faz uma alteração em um repositório, ele pode criar uma solicitação de *pull request* para enviar essas alterações para revisão. Outros desenvolvedores podem revisar o código, deixar comentários e, por fim, aprovar ou rejeitar as alterações propostas. Depois que as alterações forem aprovadas, elas poderão ser mescladas na ramificação principal do repositório.

No contexto de trabalhos de hardware, um PR pode ser usado para adicionar ou atualizar o código-fonte de um *firmware*, ou para adicionar ou atualizar os arquivos de design de um circuito. Esse processo permite que outros membros da equipe revejam e deliberem sobre a aprovação ou não das alterações sugeridas. Uma solicitação *pull* ainda pode ser usada como uma forma de os desenvolvedores proporem alterações em um projeto e colaborarem com outras pessoas no desenvolvimento do hardware.

Por exemplo, no caso de hardware eletrônico, um membro da equipe pode propor uma modificação no projeto do circuito e outros membros da equipe podem revisar as alterações propostas, deixando comentários e aprovando ou rejeitando a modificação proposta. Em geral, o uso de solicitações *pull request* no desenvolvimento de hardware pode ajudar a manter um processo de desenvolvimento consistente e organizado, permitindo uma colaboração eficiente e revisão das mudanças propostas. Também pode ser usado como uma forma de acompanhar o progresso e manter um registro das alterações feitas no design do hardware.

No entanto, é importante observar que o desenvolvimento de hardware geralmente envolve testes físicos e experimentação, o que pode não ser possível realizar em um processo de revisão de solicitação.

3.11 Colaborando na plataforma

Para fazer alguma sugestão, correção ou adicionar qualquer arquivo, é necessário entrar em contato com a comunidade, expor a proposta e caso aprovado, é possível ”Forkear” o código e iniciar o processo de colaboração, em seguida a comunidade irá avaliar e caso aprovado pode ser publicado.

A documentação para iniciar um projeto é concentrada na página da Franzini-

nho[7] em primeiros passos. No caso desse projeto, que foi aplicada a Franzininho Wifi e *Circuitpython*, a parte explorada fica na *sidebar* como é exibido na Figura 3.

Figura 3 – Fluxo da documentação

A documentação possui o básico para iniciar a utilizar-se do *Circuitpython*, indo direto ao ponto em que vamos trabalhar, de forma objetiva e clara. É importante seguirmos esse padrão. Nessa página de exemplos, tivemos a oportunidade de trazer o nosso sistema de irrigação automático desenvolvido neste estudo.

O *frontend* do site da comunidade é feito em uma plataforma chamada Docusaurus[8], esse projeto é feito para construir, implantar e manter facilmente sites de projetos de código aberto. Ele é desenvolvido pelo *Facebook* e é baseado no *React*, uma biblioteca *JavaScript* popular para criar interfaces de usuário. A plataforma fornece uma maneira simples de criar documentação, blogs e outros tipos de sites para projetos de código aberto. Ele foi projetado para ser rápido, escalável e fácil de usar, com recursos como controle de versão automático e recursos de pesquisa integrados. A plataforma também é altamente personalizável, permitindo que os usuários criem sites exclusivos e atraentes para seus projetos.

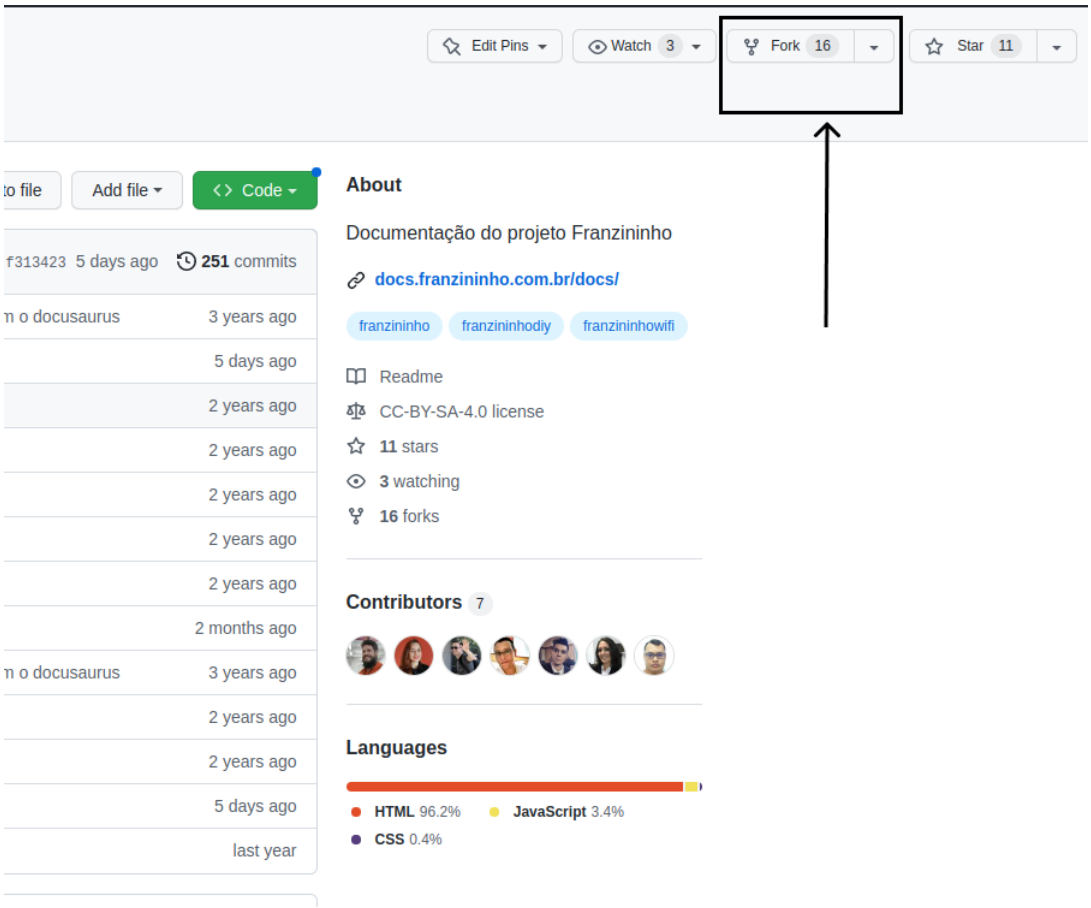
A comunidade hospeda a plataforma no *GitHub* da Franzininho, em <https://github.com/Franzininho/docs-franzininho-site>. Nessa página estão todos os detalhes de

[7] https://docs.franzininho.com.br/docs/franzininho-wifi/exemplos-circuitpython/primeiros-passos/
[8] https://docusaurus.io/pt-BR/docs

como é possível contribuir. Nesta página do *GitHub* há um fluxo de como testar e validar uma contribuição de um projeto, também fica concentrada toda a documentação para os primeiros passos, exemplos com as placas e projetos Franzininho.

Ao expormos a proposta de colaboração por e-mail, foi realizado um *meet on-line* , onde ela foi discutida e aprovada. Em seguida realizamos um *Fork* do conteúdo da página, como mostra a imagem abaixo, na Figura 4.

Figura 4 – Fork do GitHub Franzininho



Fazer um *fork* de um projeto é uma forma de trazer um repositório remoto para sua conta e personalizá-lo. A título de exemplo, ao encontrar o projeto que deseja *forkear* em páginas como *GitHub, GitLab, BitBucket* ou qualquer outro serviço de controle de versão, no caso da Franzininho é o *GitHub*, uma vez que encontrado o repositório desejado, foi clicado no botão "Fork" na interface da web e em seguida, seguimos os passos confirmando que quer uma copia na sua conta. Depois que obtivemos uma cópia desse repositório na conta desejada, clonamos o repositório com o comando abaixo, para a própria maquina:

```
$git clone https://github.com/<username>/<repository>.git
```

Substitua o <username> e o <repository> pelos detalhes do repositório que você acabou de *forkear*, no nosso caso foi o comando:

```
$https://github.com/GiulianeEC/docs-franzininho-site.git
```

Esse ambiente foi levado para o meu computador pessoal, depois de clonar o repositório, foi possível fazer a adição. O projeto é baseado em Node.js com Yarn, então foi necessário executar o comando abaixo para obter as configurações necessárias:

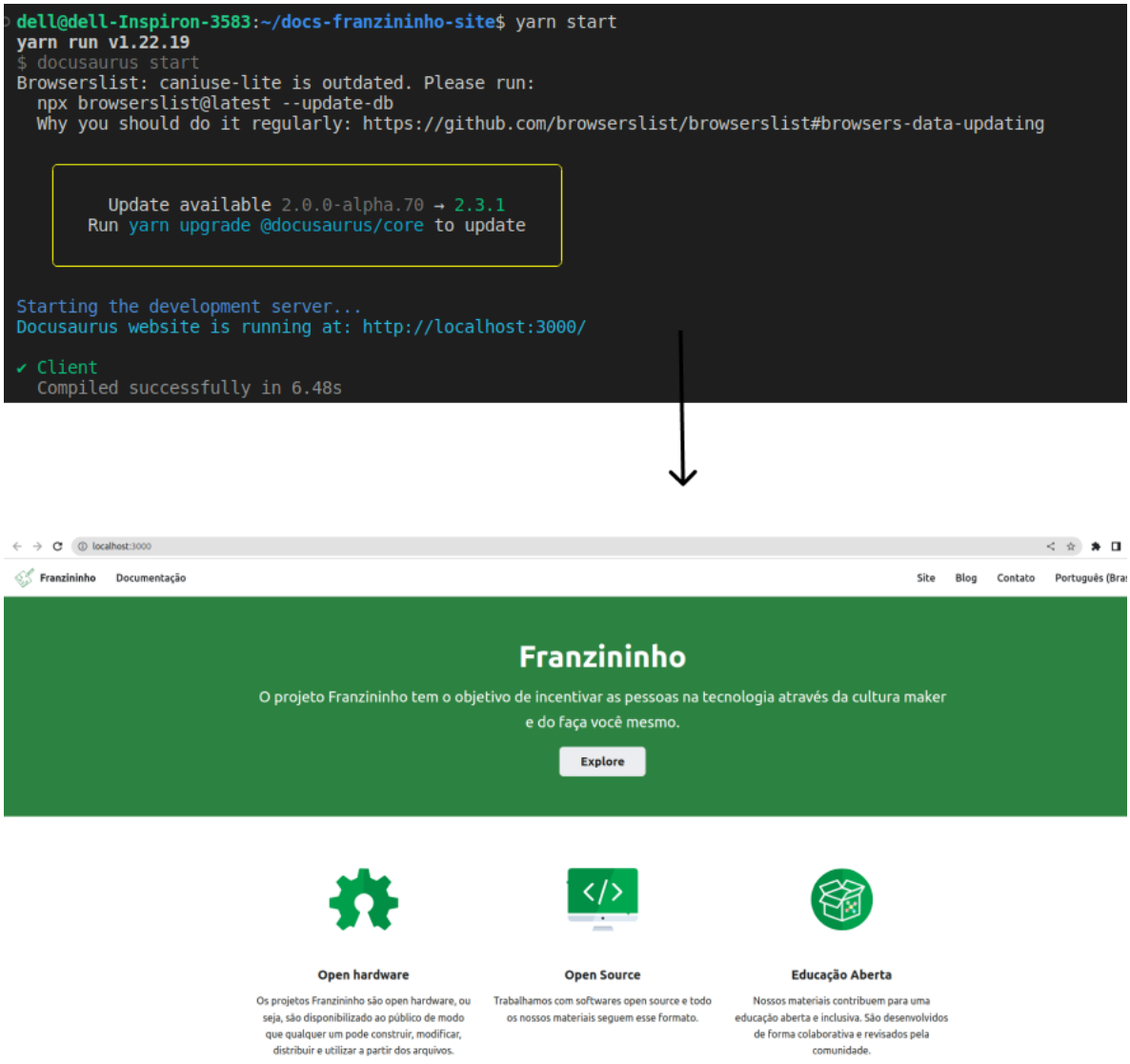
```
$yarn install
```

É possível executar o projeto de forma local. Então dentro do repositório executamos o comando para gerar um *localhost*, sendo possível acompanhar as alterações de forma visual no ambiente de desenvolvimento, acompanhando como ficará oficialmente, se está seguindo o rumo desejado de desenvolvimento e se aconteceu algum erro:

```
$yarn start
```

Toda vez que o arquivo é salvo este comando atualiza automaticamente esse *localhost* com a alteração. Na Figura 5 é possível ver como ficou no terminal do VsCode, como fica após a execução do *yarn start*, e como é o esperado a página local ser aberta, através do link <http://localhost:3000/> no navegador.

Figura 5 – Página local pelo crome e terminal do VsCode

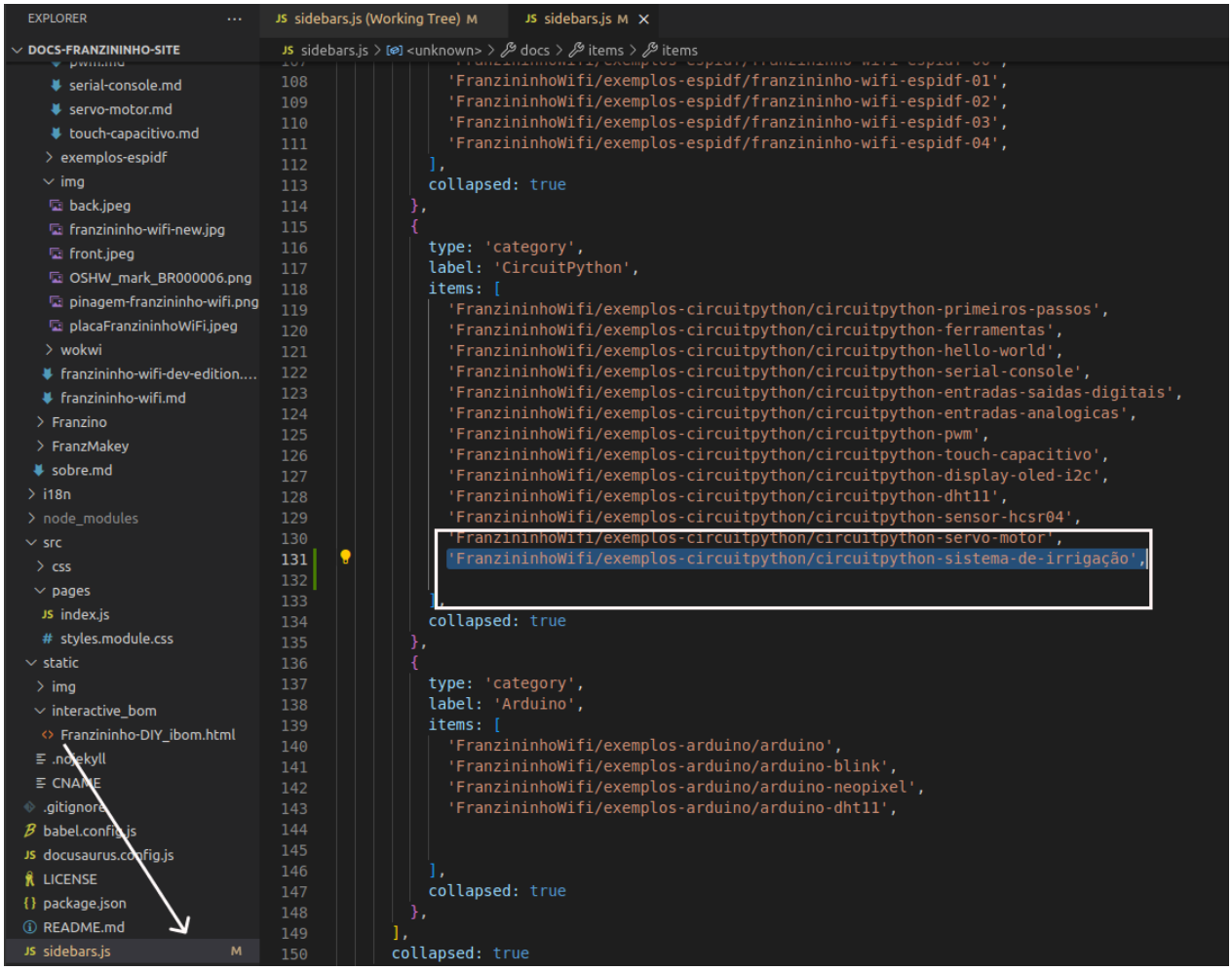


A página foi criada em FranzininhoWiFi - > exemplos-circuitpython, um arquivo irrigação.md foi criado nessa pasta e dentro dele adicionamos todo o código seguindo a proposta feita e as regras necessárias para a publicação.

Ao abrir o editor de texto, designado bastou seguir o padrão das páginas já criadas. Para criar uma página foi necessário ir em *sidebar* e adicionar o nome da página que estaria criando com o nome do diretório criado.

Em seguida será necessário fazer *checkout* para uma nova *branch*, ao criar essa *branch* foi possível fazer as edições. Em seguida fizemos um *commit* com uma breve mensagem, pontuando o que foi realizado no interativo e na terceira pessoa. Na Figura 6, é possível ver o exemplo no código de como foi adicionado uma nova página na *sidebar* nesse projeto, e o nome do novo link da nova página que foi criada.

Figura 6 – Adicionando na sidebar



Com as mudanças feitas em ambiente de desenvolvimento, testadas e conferidas (nós como desenvolvedores, somos responsáveis pelas alterações feitas e pela qualidade do software), em seguida foi realizado um *push* para o repositório e terá a opção de enviar um *pull request*, onde será avaliado e aprovado antes de passar para a documentação oficial.

Na página do *GitHub* da comunidade existe um fluxo de como funciona as etapas de colaboração. Para escrita de novos tópicos e inserção de novos textos que podem ser tutoriais ou projetos sobre algumas das placas e ferramentas Franzininho, para isso veja os seguintes passos:

- Se a proposta for aprovada, faça Fork do projeto;
- Crie uma *Issue* sobre o novo tópico e discuta sobre a necessidade e viabilidade;
- Crie uma nova branch;

- Crie uma pasta com o texto dentro da pasta contribuição, seguindo a seguinte estrutura (se preferir copie o exemplo na pasta) Contribuição:
  - 1 nome-da-pasta-com-seu-texto:
    - texto.md
    - img
      - img1.png
      - img2.png
  - 5 Edite seu texto
  - 6 Commit suas mudanças:

    ```
    $git commit -m 'Adicionei nome-do-projeto'
    ```

  - 7 Push para a branch:

    ```
    $git push origin nova-branch
    ```

- Abra um Pull Request
- Após a revisão e aprovação pelo time de documentação será feito o deploy
- Depois que o seu Pull Request for unido ao projeto, você pode deletar a sua branch.

O *GitHub* oferece diversas maneiras para contribuir para os projetos do Franzininho. Contribuidores podem realizar tarefas de codificação, realizar testes, melhorar a documentação ou sugerir recursos e melhorias para os projetos. Os contribuidores também podem ajudar a resolver problemas e responder às perguntas relacionadas aos projetos da comunidade.

Como principal repositório de código aberto, o *GitHub* fornece ferramentas para facilitar o desenvolvimento de projetos, bem como o compartilhamento de código entre os membros da comunidade. Além disso, o *GitHub* fornece diversas maneiras de promover os projetos da comunidade Franzininho, como a criação de guias, tutoriais e outros conteúdos relacionados.

Os membros da comunidade também podem fazer uso do *GitHub* para criar e gerenciar projetos colaborativos, obter apoio da comunidade para seus próprios projetos e ajudar outros a se envolverem com o Franzininho. Basicamente, esse é o fluxo para contribuir com um tutorial de algum projeto ou instalação realizada para a Franzininho. A comunidade é bem ativa no *Telegram* e *e-mails*, o que facilitou todo o processo na hora de colaborar.

3.12 Pull request da página do sistema de irrigação

Ao realizar os passos citados no capitulo 3.11, foi desenvolvido uma página local na minha máquina, de um tutorial com sugestões de como montar um protótipo de um sistema de irrigação automático empregando a placa franzininho WiFi com *Circuitpython* na IDE Mu-Editor.

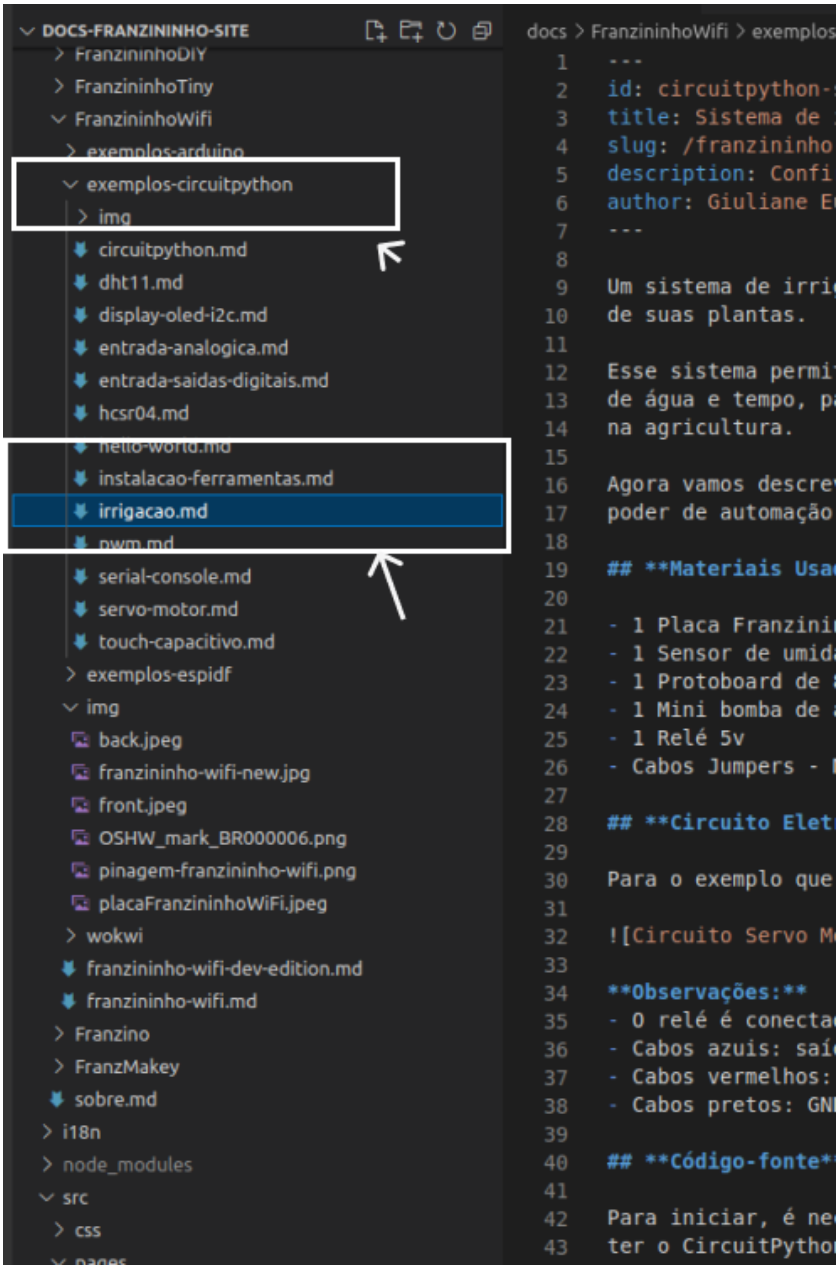
Como é clonado um ambiente, todas alterações feitas ficam no ambiente de desenvolvimento, onde podem ser testadas, analisadas e refeitas quantas vezes necessárias antes de enviar a *branch* criada, fazer um *push* para a *master* e solicitar uma análise da comunidade.

Para criar uma *branch*, após fazer um *Fork* no GitHub pelo link e abrir o ambiente local, como é explicado no cap 3.10, para empenhar o comando abaixo. O nome da *branch* fica ao seu critério, desde que seja um nome que fale sobre o que será desenvolvido:

```
$git checkout -b irrigação-TCC-Giuliane
```

Dentro da *branch* criada, foi inserido um arquivo *irrigacao.md* onde foi inserido toda a construção da página. Foi optado pelo *VsCode* para fazer essa construção, na Figura 7 abaixo é possível ver o local e o arquivo criado como exemplo. Observamos como a estrutura de um tutorial é desenvolvida dentro do site, para ir seguindo como exemplo, buscamos fazer um jeito autêntico mas sem fugir do padrão. Na primeira parte foi criada uma explicação e o propósito do protótipo, em seguida foram listados os materiais utilizados e um modelo de como o sistema fica montado quando pronto. Por ultimo foi exposto código fonte e a explicação de cada funcionalidade e como ela interage no código-fonte.

Figura 7 – Arquivo criado em exemplos-circuipython no ambiente de desenvolvimento



Após verificar, e ver se está coerente com a ideia inicial criada sobre o tutorial, foi possível subir as alterações para o *GitHub* para a comunidade avaliar, solicitar mudanças ou rejeitar.

Os comandos necessários para subir os arquivos foram :

```
$ git status
$ git add .
$ git status
$ git commit -m "TCC Giuliane - sistema de irrigação"
```

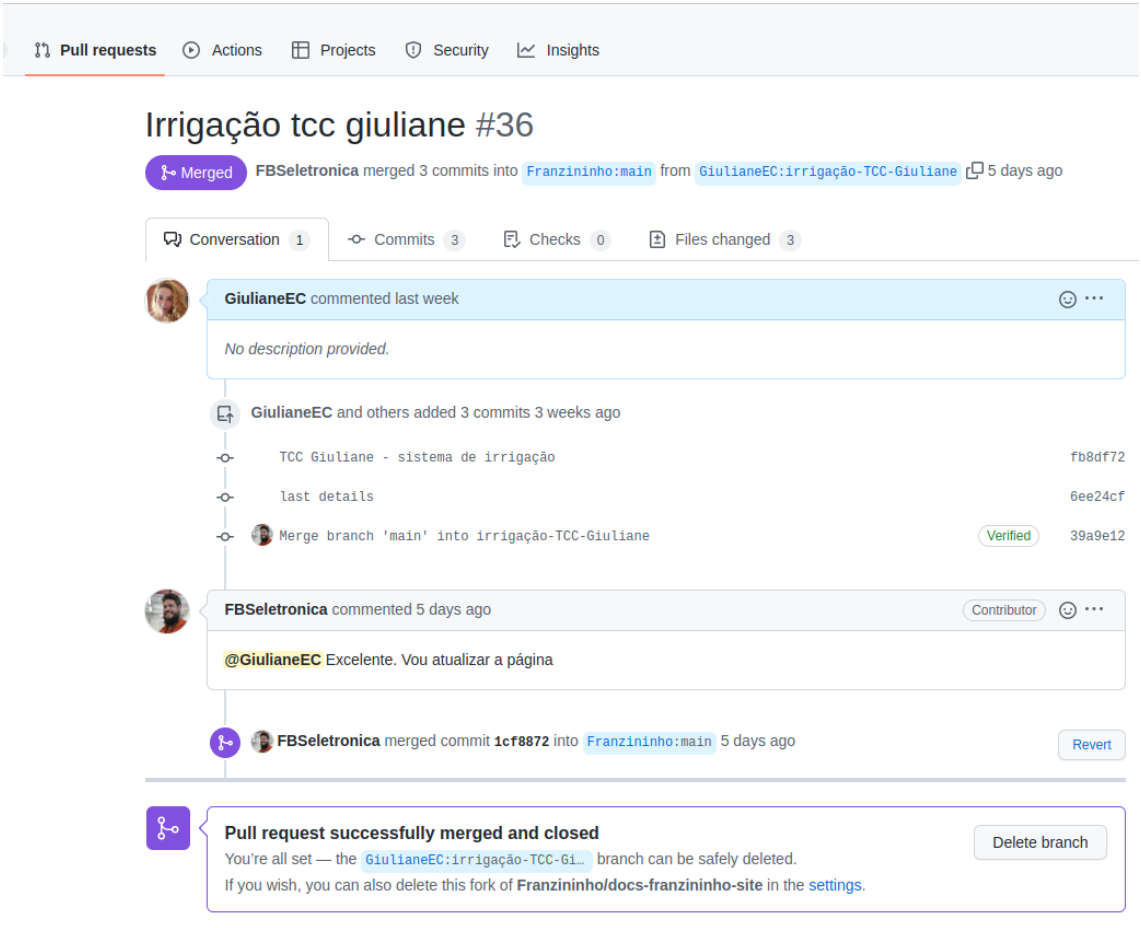
O comando *"git status"* é útil para acompanhar as mudanças realizadas no arquivo, ver o status dos arquivos editados ou criados. O *"git add ."* adiciona as alterações feitas e os arquivos modificados. Após a execução desses comandos, podemos realizar o *commit*, através do *git commit -m* e escrever a mensagem que representa as alterações.

Após os arquivos serem alterados, adicionados e feito o commit. Foram mergeados a master na nossa branch pelo *git merge master*. Em seguida ao ir a master, podemos fazer o *push* pelo comando *git push origin*, como é possível ver abaixo:

```
$ git merge master
$ git checkout master
$ git push origin master
```

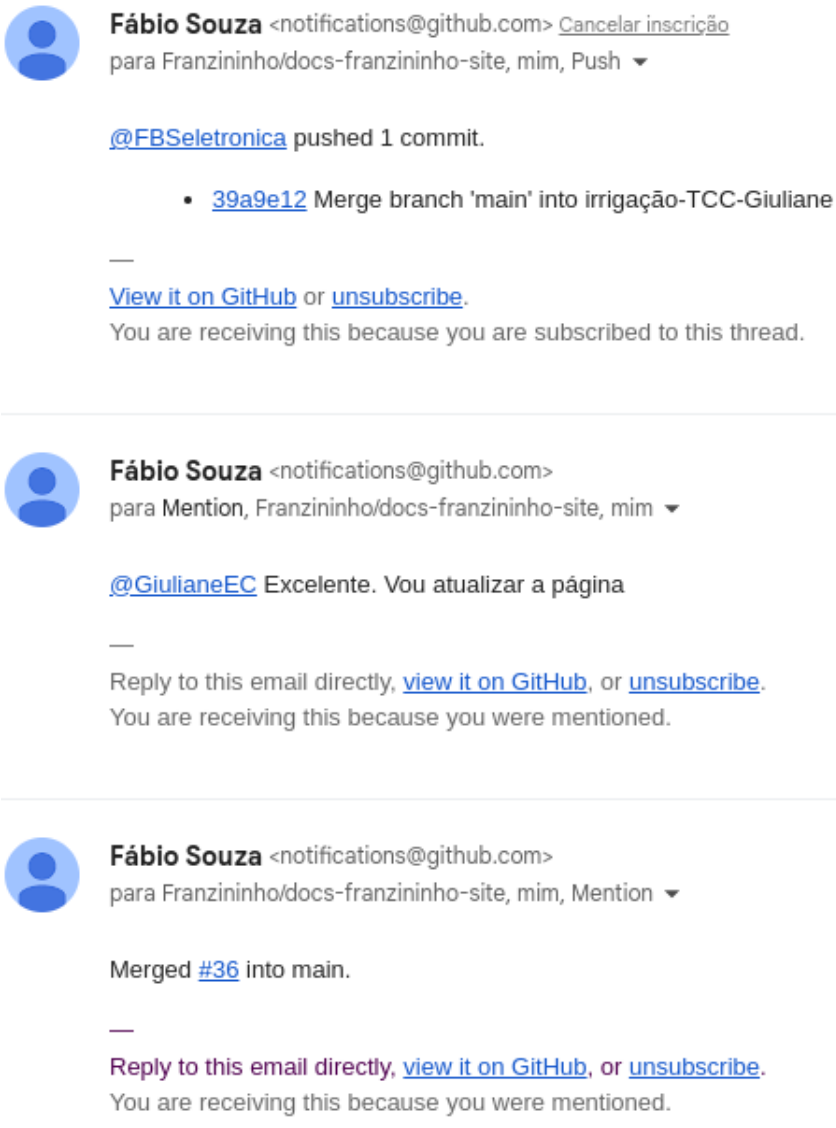
Por fim, é possível no *GitHub*, acompanhar o status do *pull request*, na Figura 8 está todas atualizações, e como foi aprovado, mergeado na página e publicado na página oficial do Franzininho. Esse status também podem ser acompanhados pelo <https://github.com/Franzininho/docs-franzininho-site/pull/36>.

Figura 8 – imagem da tela do GItHub para acompanhar as solicitações



O *GitHub* também nos notifica por *e-mail* as atualizações sobre a solicitação como podemos ver na Figura 9. Após a aprovação e validação da página web sobre o protótipo do projeto de irrigação, a página foi publicada na página oficial, e pode ser conferida através do link: <https://docs.franzininho.com.br/docs/franzininho-wifi/exemplos-circuitpython/irrigacao>

Figura 9 – E-mails enviados pelo GitHub do status das solicitações



4 Resultados

Para a elaboração do estudo, foi importante ter conhecimento prévio de programação e eletrônica, bem como ter familiaridade com o funcionamento de sensores e atuadores. Além disso, foi importante realizar testes no sistema de irrigação automático para garantir seu funcionamento e certificar de que não havia falhas no mesmo. A experiência de montar um sistema de irrigação automático com *CircuitPython* pode ser desafiadora, mas após o primeiro contato com a placa e com as oportunidades que ela explora, o uso se torna cativante.

Os resultados desses estudos mostram convergência e divergência no estado da arte e na prática do desenvolvimento. O estudo sobre "O que acontece quando o bazar cresce", citado na revisão bibliográfica, aborda as divergências na forma como as informações são apresentadas e organizadas, como as estatísticas sendo agrupadas por ano em estudos acadêmicos, enquanto são apresentadas por liberação do *kernel* em publicações da comunidade. Neste trabalho foi possível ver dificuldades semelhantes, principalmente pela falta de estudos sobre hardwares livres e também pela falta de pessoas capacitadas envolvidas em projetos ou comunidades para buscar referências.

Encontrar uma tarefa inicial, configurar o espaço de trabalho local, a falta de envolvimento inicial com a comunidade, expectativas pouco claras e informações sobrecarregadas, além do medo de cometer erros por ser uma recém-chegada a projetos *OSH*, também não poder ter tido contato com outras mulheres que realizaram projetos, nesse seguimento de colaboração com comunidades, foram algumas barreiras e dificuldades encontradas na elaboração do projeto.

Diante disso, se fez necessário buscar ajuda para obter *insights* sobre maneiras de facilitar esse início. Com o objetivo de pesquisar sobre a comunidade, construir um projeto para analisar a conscientização e mapear as barreiras das ferramentas existentes de estudo e na prática, foi essencial pesquisar e ter conversas constantes com os membros principais em projetos OSH, para confirmar as descobertas e explorar as barreiras com mais profundidade. A maior parte dessa interação foi realizada através de *e-mail* e *Telegram*.

Um dos desafios enfrentados, foi não ver mulheres no ambiente. Ás vezes pode ser desafiador não ter contato com modelos femininos, como mentoras ou de apoio. A indústria de software atualmente lida com essa realidade de pouca representatividade feminina, além de alguns problemas como disparidade salarial entre homens e mulheres, oportunidades limitadas de progresso e sexismo. Tudo isso faz o ambiente de trabalho se tornar mais desafiador, pelo fato de existir poucos exemplos femininos na indústria de software e hardware.

Apesar de não ter sofrido com os desafios de desigualdade de gênero nesse trabalho, Trinkenreich (2022) cita que é possível tirar conclusões que os desafios encontrados na indústria de software e hardware são semelhantes aos desafios enfrentados pelas mulheres em outras indústria predominadas por homens, esses desafios incluem preconceito de gênero, discriminação e assédio. É preciso usar estrategias para superar desafios como *networking*, orientação e autopromoção.

Foi possível aprender com a comunidade, e ver que o mercado está evoluindo cada vez mais. Observamos que para começar basta ter a iniciativa, mesmo que pequeno, esse grupo de pessoas apaixonadas pela plataforma irá te oferecer suporte e te ajudará a contribuir com as ideias propostas, além de ser uma boa maneira de obter *feedback* e outras ideias de usuários.

A Franzininho pôde nos mostrar a importância de promover a colaboração e o compartilhamento, incentivar os membros da comunidade a compartilhar seus projetos, ideias e experiências uns com os outros pode ajudar a criar um sentimento de propriedade e pertencimento, e também levar a inovações e *insights*.

O fornecimento de recursos e suporte, nos ajudou a certificarmo-nos de que caso exista o desejo de se participar de uma comunidade, essa comunidade contará com acesso aos recursos e suporte necessários. Isso inclui documentação técnica, tutoriais, fóruns e outros recursos. E por fim, e mais importante, o incentivo a participação e a transparência do processo de criação. Junto com o incentivo dos membros de comunidade, a se envolverem no desenvolvimento e na evolução da plataforma.

Poder incluir e contribuir com o tutorial, escrever documentação e ajudar a testar novos recursos, reforça na prática toda teoria estudada nesse trabalho. O fato de ser aberto e transparente sobre o processo de desenvolvimento ajudou a construir confiança e promover um senso de colaboração sem muitas barreiras e burocracias, com o principal objetivo ser colaborar e agregar com novas experiências e conhecimento.

5 Conclusão

Como contribuidora e membra da comunidade, foi possível obter uma experiência positiva ao trabalhar e utilizar a placa da comunidade de hardware livre Franzininho, para criar um sistema de irrigação automático.

É muito satisfatório saber que a comunidade oferece suporte e recursos para iniciantes, possibilitando me tornar uma colaboradora da mesma. O fato da comunidade estar ativa nas redes sociais, oferecer recursos como vídeos no *YouTube* e canais de interação para tirar dúvidas, pelo *Telegram*, *Discord* e *e-mail*, é muito útil para as pessoas que estão começando a trabalhar com hardware livre. É importante ressaltar que o hardware livre é uma alternativa ao modelo tradicional de propriedade intelectual, podendo ser uma forma de promover a colaboração e o compartilhamento de conhecimento e experiência na comunidade de tecnologia.

A comunidade Franzininho demonstrou ser uma ótima fonte de apoio para iniciantes com hardware de código aberto. É importante ter acesso a documentação e tutoriais para poder iniciar um projeto com sucesso. Além disso, ter canais de interação com a comunidade é uma ótima forma de obter ajuda quando surgem dúvidas ou problemas durante o processo de desenvolvimento.

Neste estudo de caso, observou-se como o hardware e software aberto podem fornecer vantagens para trabalhos futuros, oferecendo muitos benefícios ao permitir que as pessoas acessem e modifiquem o código e o tamanho do protótipo com funcionalidades, como por exemplo; poder ser expandido para controle, ser monitorado via *app*, criar uma interface agradável ao usuário de monitoramentos, aumentar o poder e tamanho com mais atuadores e dispositivos IoT, criar softwares embarcados e soluções de computação para agregar na placa e com a comunidade. Uma das dificuldades foram encontrar documentos e materiais exemplos de estudos de caso voltados para *open hardware* comparado ao peso de *open software* nos estudos, isso também pode ser abordado como trabalho futuro.

Acredito que realizar um estudo sobre a quantidade de mulheres ativas nessa área, tendo em vista que é de predominância masculina, é um estudo que vale a pena ser explorado, buscando soluções para igualar ou aproximar os números, em relação ao gênero na área. Mesmo que atualmente existe uma pequena porcentagem feminina, assumindo vagas em estudos e experimentos em hardwares, a discrepância ainda incomoda. A rotulação de cargos por gênero já é algo que está fora de tendencia, não existe cargo para homem e cargo para mulher, existem cargos que podem ser assumidos por qualquer pessoa que deseja e esteja qualificada para isso.

Essa pesquisa me permitiu explorar a parte prática e teórica do estudos de software

com foco em hardware. Conhecer a comunidade de perto, ouvir, abstrair e finalizar com a colaboração na página oficial, permitiu quebrar algumas barreiras e crenças sobre a burocracia, de que pode ser difícil o processo e cheio de etapas complexas. O conceito *open* está buscando mostrar cada vez mais que com menos burocracia, mais coisas podem ser desenvolvidas, compartilhadas e expandidas. Evitando o processo de engenharia reversa para descobrir o caminho e depois trazer a inovação. Com o caminho aberto e o material disponível, precisamos aplicar ideias novas e transformações. Esse poder pode fazer toda a diferença no mundo da engenharia, tecnologia e inovação.

Por fim, fica o incentivo para poder ingressar em uma comunidade e se beneficiar dos conhecimentos por ela oferecidos, colaborar e ajudar o modelo de conhecimento aberto a crescer cada vez mais. Acredito também no poder de construir novas comunidades para abranger e expandir cada vez mais esse movimento no mercado. Construir uma comunidade de criadores em torno de uma plataforma de hardware aberta pode ser uma ótima maneira de promover inovação, colaboração e aprendizado.

Referências

FRONCHETTI, F. What attracts newcomers to onboard on oss projects? tl;dr: Popularity. In: . USA: INorthern Arizona University, Flagstaff,Arizona, USA, 2013. Disponível em: <https://www.researchgate.net/publication/333319625_What_Attracts_Newcomers_to_Onboard_on_OSS_Projects_TLDR_Popularity>. Acesso em: 03 MAR. 2023. Citado na página 19.

INTELLIGENCE, M. *Internet of Things (IoT) Market - Growth, Trends, COVID-19 Impact, and Forecasts*. 2022. Url: https://gr1d.io/2022/09/02/inteligencia-artificial-3/ . Citado na página 16.

MORREALE1, F. Building a maker community around an open hardware platform. In: *Centre for Digital Music1 Mixed Reality Laboratory2 School of EECS*. London: Mixed Reality Laboratory, 2017. Disponível em: <https://www.slideshare.net/FabioMorreale1/building-a-maker-community-around-an-open-hardware-platform>. Acesso em: 03 out. 2022. Citado na página 18.

ORACLE. *O que é IoT?* 2017. Disponível em: <https://www.oracle.com/br/internet-of-things/what-is-iot/>. Citado na página 15.

POZZEBOM, R. *O que são sistemas embarcados?* 2021. Disponível em: <https://www.oficinadanet.com.br/post/13538-o-que-sao-sistemas-embarcados>. Citado na página 15.

RAYMOND, E. S. *The Cathedral and the Bazaar*. New York, NY, USA: O'Reilly Media, 1999. Citado na página 18.

SANTOS, B. P. *Internet das Coisas: da Teoria à Prática*. 2021. Https://homepages.dcc.ufmg.br/ mmvieira/cc/papers/internet-das-coisas.pdf. Citado na página 25.

SMITH, B. Gplv3. In: . USA: [s.n.], 2022. Disponível em: <https://www.gnu.org/licenses/quick-guide-gplv3.html>. Acesso em: 03 MAR. 2023. Citado na página 21.

SOUZA, F. *Open source hardware: Conheça a definição e as boas práticas*. 2019. Urlhttps://embarcados.com.br/open-hardware-definicao/. Citado na página 21.

STEINMACHER, I. Why do newcomers abandon open source software projects? In: . CA, USA: International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), San Francisco, 2013. Disponível em: <doi:10.1109/CHASE.2013.6614728.> Acesso em: 03 MAR. 2023. Citado na página 19.

TRINKENREICH, B. *An Empirical Investigation on the Challenges Faced by Women in the Software Industry: A Case Study*: um estudo abrangente sobre o modelo contemporâneo de desenvolvimento do kernel linux. Dissertação (Mestrado) — Cornell University, NY, mar. 2022. Citado 2 vezes nas páginas 19 e 45.

WEN, M. S. R. *O que acontece quando o bazar cresce*: um estudo abrangente sobre o modelo contemporâneo de desenvolvimento do kernel linux. Dissertação (Mestrado) — Universidade de São Paulo, São Paulo, mar. 2021. Citado na página 19.

Anexos

ANEXO A – Outros materiais

Figura 10 – Protótipo em atuação

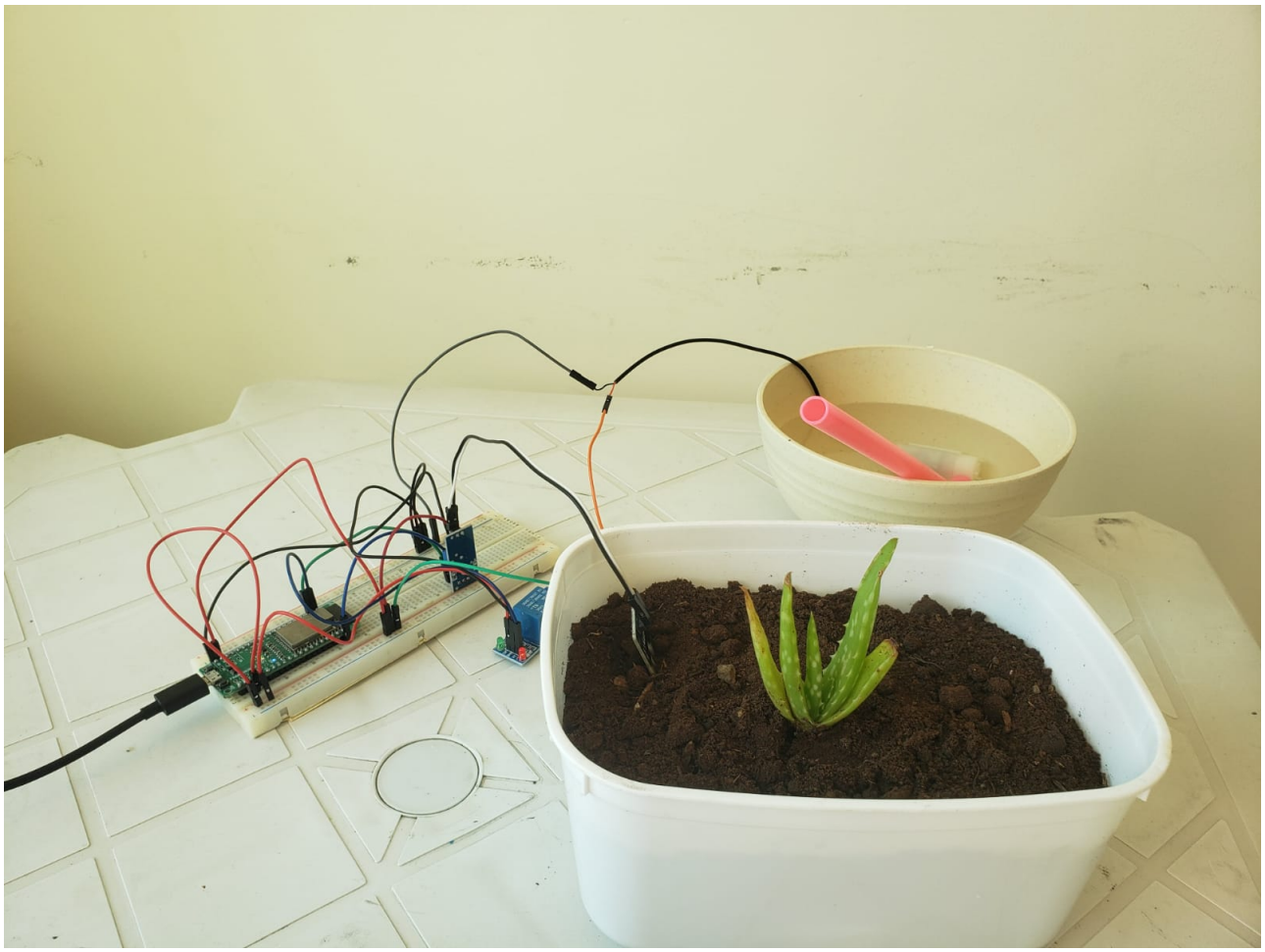


Figura 11 – Protótipo em atuação 2