Ana Emília Hernandes Dib

Projeto de irrigação de jardim utilizando Internet das Coisas

São Paulo, SP

2020

Ana Emília Hernandes Dib

Projeto de irrigação de jardim utilizando Internet das Coisas

Trabalho de formatura apresentada à Escola Politécnica da Universidade de São Paulo como parte dos requisitos para obtenção do grau de Bacharel em Engenharia Mecatrônica.

Universidade de São Paulo

Escola Politécnica

Orientador: Prof. Dr. Fabrício Junqueira

São Paulo, SP

2020

Catalogação-na-publicação

Dib, Ana Emília Hernandes
Projeto de irrigação de jardim utilizando Internet das Coisas / A. E. H. Dib - São Paulo, 2020.
80 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Internet das Coisas 2.Jardim 3.Automação residencial I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t.

Agradecimentos

Ao meu orientador Fabrício, por toda orientação, paciência e ajuda. Aos meus pais pelo suporte durante todos os anos da graduação.

Ao Nilton, ao Daniel e principalmente ao Luciano pelo apoio e compreensão durante esse tempo. Ao Lucas pela experiência transmitida e também pelos conselhos oferecidos para a realização deste trabalho.

Resumo

A Internet das Coisas é um termo utilizado para se referir à conexão de objetos "comuns"do dia-a-dia à internet. No contexto de Indústria 4.0, existe cada vez mais uma tendência a objetos e máquinas operarem conectados, com informações em tempo real. Neste trabalho, é proposto e desenvolvido um projeto de automação residencial, de sistema autônomo de irrigação de jardins, utilizando um Raspberry Pi modelo B+ e um módulo NodeMCU com microcontrolador ESP8266. O sistema é capaz de se conectar à internet para obter a previsão do tempo e fazer comunicação entre seus componentes, e utiliza sensor de umidade e fluxo para verificar a umidade do solo e medir o volume de água utilizado, respectivamente. Além disso, o sistema oferece a possibilidade de operação automática, em que faz irrigações periódicas constantes, independente da previsão do tempo. Após levantados os requisitos de projeto, o sistema foi modelado a partir de diagramas UML e implementado em três frentes: controlador, processador e interface com usuário. Após implementado, o sistema foi colocado em execução, e obteve resultados consistentes com o objetivo esperado.

Palavras-chave: Internet das Coisas. Jardim. Automação residencial.

Abstract

Internet of Things is a term used to refer to the connection of "common" everyday objects to the internet. In the context of Industry 4.0, there is an increasing tendency for objects and machines to operate connected, with real-time information. This monography proposes and develops a home automation project, with an autonomous garden irrigation system, using a Raspberry Pi model B + and a NodeMCU module with ESP8266 microcontroller. The system is capable to connect to the internet to get the weather forecast and to communicate between the components, and uses the humidity and flow sensors to verify the soil moisture and measure the volume of water used, respectively. In addition, the system offers the possibility to operate in automatic mode, in which it makes constant periodic irrigations, regardless of the weather forecast. After defining the project requirements, the system was modeled using UML diagrams and implemented based on three fronts: controller, processor and user interface. After implemented, the system was put into execution, and the results obtained were consistent with the objective expected.

Keywords: Internet of Things. Garden. Home automation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
AWS	*Amazon Web Services*
ID	Identificação
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IBM	*International Business Machines*
IoT	*Internet of Things*
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
JSON	*JavaScript Object Notation*
MQTT	*Message Queuing Telemetry Transport*
NBR	Norma Brasileira
NTP	*Network Time Protocol*
OASIS	*Organization for the Advancement of Structured Information Standards*
PIB	Produto Interno Bruto
PHP	*Hypertext Preprocessor*
SSID	*Service Set Identifier*
TCP	*Transmission Control Protocol*
UML	*Unified Modeling Language*

Sumário

1 Introdução

A irrigação é um meio artificial de fornecer água ao solo para o cultivo de plantas (RODRIGUES; SOUSA, 2018). Ela é necessária para proporcionar a quantidade adequada de água ao plantio, complementando a água proveniente das chuvas. A irrigação visa otimizar os recursos hídricos disponíveis para cada região e situação.

De acordo com Neto (2017), essa prática é utilizada desde 4500 a.C. Diversas civilizações nasceram e se desenvolveram graças à irrigação. O primeiro aspersor giratório foi desenvolvido em 1926, para aplicação em irrigação de jardim. A invenção do aspersor de impacto em 1933 gerou uma revolução nos métodos de irrigação no mundo. A Figura 1.1 mostra a patente registrada desse aspersor. Atualmente, o uso da irrigação é feito para garantir a produção agrícola em regiões com baixo índice pluviométrico, e aumento da produtividade das culturas e qualidade dos produtos Testezlaf (2017).

Para Lucon e Chaves (2004), a irrigação também é uma prática importante no cultivo de hortas, que ganha espaço desde a década de 1970, como uma resposta ao uso abundante de agrotóxicos e conscientização ambiental. Nessas hortas a irrigação é fundamental para o desenvolvimento das plantas.

Segundo Gengo e Henkes (2013), nas cidades e áreas urbanas, jardins são usados como um instrumento de gestão ambiental, para melhoria de qualidade de vida, ambiental e estética urbana. Além disso, o paisagismo é uma solução que vem sendo usada nas empresas para se adequarem a normas ambientais, como a NBR ISO 14001. O setor de turismo também investe em paisagismo, visto que os clientes tendem a optar por espaços sustentáveis. A irrigação se faz necessária na manutenção e prosperidade desses jardins.

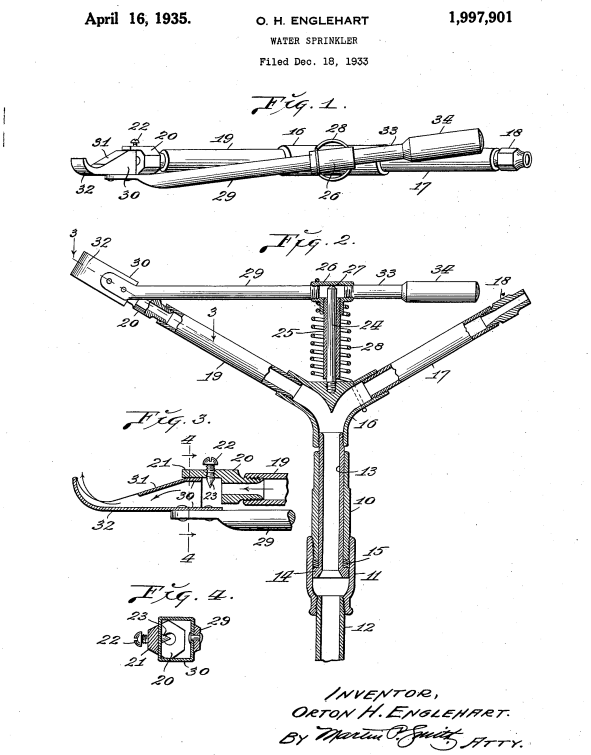
Ao passo em que novas tecnologias são desenvolvidas, cresce a demanda de automação e, consequentemente, a aplicação do uso da internet das coisas em objetos do dia-a-dia. Desse modo, crescem também a quantidade de aplicações disponíveis para facilitar essa conexão das coisas por meio da internet.

No mercado existem produtos[1] destinados à irrigação de jardim, de topo de linha, baseados em *IoT*, que se conectam à rede *WiFi* e permitem que o usuário controle o horário e duração da irrigação à distância, por meio de aplicativo de celular. Há produtos que possuem também inteligência climática, que ajusta os cronogramas de rega baseado no clima e tipo de solo locais.

No meio acadêmico, pode-se encontrar diversas abordagens de aplicação de irrigação inteligente de jardins. Na década passada Santos (2010) desenvolveu um sistema de controle

[1] *Rachio Smart Sprinkle Controller* e *Orbit B-Hyve*, vendidos pela *Amazon*.

Figura 1.1 – Patente US1997901A



Fonte: Google Patents. Disponível em <https://patents.google.com/patent/US1997901A/en>. Acesso em 16 jun. 2020.

de irrigação automático, porém sem utilizar conceitos de Internet das Coisas. Seu projeto foi baseado em sensor de temperatura LM35[2], e se considerou que, em um dia de chuva, a temperatura ambiente seria mais baixa, deste modo o controle é feito quando a temperatura atinge um valor mínimo determinado pelo autor.

Para a irrigação, o microcontrolador PIC18F452[3] aciona uma bomba por um intervalo de tempo. O autor destacou que é importante observar o clima local antes de configurar o sistema. Apesar de concluir o projeto, a plataforma experimental não foi posta em prática.

Ghizzi (2016) propôs um sistema de irrigação de jardim residencial automatizado. Seu objetivo principal foi a economia de água. O sistema funciona com base em sensores de umidade e na previsão de chuva local, de fontes distintas da internet. Caso a leitura do

[2] *Datasheet* disponível em: <http://www.ti.com/lit/ds/symlink/lm35.pdf>. Acesso em 18 mar. 2020.
[3] *Datasheet* disponível em: <https://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf>. Acesso em 18 mar. 2020.

sensor seja baixa, o sistema busca a informação do tempo para decidir se a irrigação é necessária ou não. Por meio de sensor de chuva é possível medir a confiabilidade de cada fonte de previsão do tempo acessada.

Além disso, o autor projetou também um sistema de coleta de águas pluviais para ser utilizado junto aos bombeamento de água, que faz parte do sistema físico do projeto Ghizzi (2016). O trabalho no todo, porém, é majoritariamente teórico, abordando com mais atenção o modelo matemático e as simulações feitas. Ao final, foi feita uma modelagem satisfatória, de acordo com o julgamento do autor, porém o modelo não foi implementado.

Em uma abordagem diferente, Correia, Rocha e Rissino (2016) apresentam uma proposta de protótipo de monitoramento remoto e controle de irrigação, calculando a quantidade de água necessária na rega a partir da evapotranspiração do solo, ou seja, a perda simultânea da água do solo por evaporação e transpiração. O cálculo da evapotranspiração possui como parâmetro a temperatura, desse modo justifica o uso deste tipo de sensor no projeto. A rega, caso necessária, é feita em horário pré-determinado pelo usuário.

O projeto de Correia, Rocha e Rissino (2016) utilizou a plataforma Arduino[4] para fazer o controle e monitoramento do sistema, com acionamento à distância via internet. Como o Arduino não possui módulo de conexão à internet, foi utilizado o circuito integrado HLK-RM04[5] para fazer essa comunicação, usando protocolos TCP/IP e HTTP. Após a fase de testes os autores concluíram que a interface do aplicativo WEB é intuitiva aos usuários, e que o sistema foi capaz de reduzir em 25% o consumo de água em seu modo automático.

Já Grehs (2016), em sua pesquisa, desenvolveu um sistema doméstico de irrigação baseado em dispositivos que medem a umidade do solo e controlam a liberação de água. As informações lidas pelos sensores são então enviadas à nuvem. O usuário, por meio de aplicativo de celular, tem acesso a essas informações e consegue monitorar e controlar o valor de umidade limite para o qual deseja a liberação de água pelos atuadores.

Na implementação do projeto, o autor priorizou o quesito econômico, focando no baixo custo. Sendo assim, utilizou um sensor de umidade do tipo HL-69[6], e uma válvula de solenoide, conectados a um microcontrolador ESP8266[7]. O protocolo utilizado para comunicação entre microcontrolador e internet foi o MQTT[8]. As informações de leitura ficam disponíveis na nuvem e ao usuário por meio do aplicativo desenvolvido.

---

[4] Mais informações em: <https://www.arduino.cc/.>

[5] *Datasheet* disponível em: <https://e-radionica.com/productdata/2013042218402981701.pdf>. Acesso em 18 mar. 2020.

[6] Mais informações em: <http://www.fecegypt.com/uploads/dataSheet/1480854383_water%20and%20soil.pdf>. Acesso em: 18 mar. 2020.

[7] *Datasheet* disponível em: <http://www.electroschematics.com/wp-content/uploads/2015/02/esp8266-datasheet.pdf>. Acesso em: 19 mar. 2020.

[8] Documentação disponível em: <http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf>. Acesso em 19 mar. 2020.

O funcionamento do projeto foi validado a partir de testes realizados. A única ressalva foi que o aplicativo desenvolvido não é intuitivo ao usuário. O autor vê a possibilidade de seu protótipo se transformar em um produto, adicionando-se protocolos de segurança, outros sensores para melhorar a precisão e uma interface amigável (GREHS, 2016).

Em seu artigo, Gonçalves et al. (2018) desenvolvem um projeto parecido com o de Grehs (2016), um sistema baseado em *IoT* que realiza irrigação automática, controlando tempo de irrigação. A rega é ativada de acordo com limites definidos para umidade do solo. No trabalho em questão, optou-se por desenvolver uma página na internet, em PHP, onde o usuário tem acesso ao monitoramento da umidade do solo.

O microcontrolador utilizado foi o NODECMU[9], que possui também o módulo ESP8266 integrado, que controla uma bomba de água. A bomba é acionada quando o sensor lê o limite mínimo e desligada no limite máximo de umidade, pré estabelecidos pelos autores. Os dados são enviados pela internet a uma página por meio do protocolo HTTP.

Gonçalves et al. (2018) sugerem que, para melhora do projeto, o microcontrolador deve ser capaz de armazenar temporariamente as informações, para evitar falhas de comunicação e perda de dados. O projeto funcionou de acordo com o esperado, podendo ser aplicado em diversas situações de irrigação.

Considerando a tendência de as pessoas se preocuparem com sua saúde e buscarem alimentos livres de agrotóxicos, Marino, Vasconcelos e Moraes (2017) desenvolveram um ambiente de irrigação inteligente de plantas para jardim que é ativado de acordo com a necessidade e monitorado à distância. Seu funcionamento é baseado em leitura de sensor de umidade para decidir se o jardim deve ser irrigado ou não.

A arquitetura inicial do projeto consistiu em utilizar um Arduino UNO[10], posteriormente substituído por um microcontrolador NODEMCU, pois este possui módulo *WiFi* integrado. O *hardware* faz a leitura dos sensores e ativação da bomba d'água. As informações obtidas são transmitidas, por meio do protocolo MQTT, a um Raspberry Pi[11], que faz o tratamento de dados e se comunica com os serviços da Amazon (AWS)[12]. Estes serviços enviam notificações por *e-mail* para o usuário.

Após diversos testes e validações, os autores concluíram que o ambiente desenvolvido desempenhou o papel proposto satisfatoriamente, e destacam que os serviços da *Amazon* são

[9] Documentação disponível em: <https://nodemcu.readthedocs.io/en/master/>. Acesso em: 05 mai. 2020

[10] *Datasheet* disponível em: <https://www.farnell.com/datasheets/1682209.pdf>. Acesso em:19 mar. 2020.

[11] *Datasheet* disponível em: <https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM3plus_1p0.pdf>. Acesso em 19 mar. 2020.

[12] *Amazon Web Services*. Mais informações em: <https://aws.amazon.com/>. Acesso em 19 mar. 2020.

promissores na área de Internet das Coisas. Por fim, reconhecem que para ser transformado em produto, devem ser estudadas a autenticação e criptografia das mensagens trocadas via internet Marino, Vasconcelos e Moraes (2017).

É válido ressaltar que a inovação científica e tecnológica está em contínuo desenvolvimento, e isso faz com que sempre surjam novas aplicações de diferentes tecnologias no mercado.

Na literatura, existem diferentes propostas e projetos de sistemas automáticos de irrigação de jardins baseados em conceitos de Internet das Coisas. Pode-se concluir que para gerar resultados, uma tecnologia pode ser aplicada de diversas maneiras, e seu uso é sempre capaz de inovar.

A Internet das Coisas é um assunto que surgiu há 20 anos e até hoje está em destaque, e possui diversas aplicações visando maior conforto, segurança e confiabilidade à população.

Na área residencial, os usuários são capazes de controlar e monitorar o ambiente remotamente, incluindo seu consumo de água. A aplicação dessas tecnologias permite otimizar o consumo e cortar gastos desnecessários.

É válido destacar que não existe um padrão de uso para esta tecnologia, que pode sempre se inovar.

1.1 Objetivos

O objetivo deste trabalho é projetar e construir um sistema de irrigação de jardins autônomo utilizando o conceito de Internet das Coisas, ou seja, de que um objeto comum do dia-a-dia tenha acesso à internet e possa se comunicar e compartilhar dados disponíveis na rede com outro objeto. Nesse caso, necessita dos dados referentes à previsão do tempo, provenientes da rede.

Para alcançar o objetivo principal, o sistema deve levar em conta a leitura da umidade do solo e a previsão do tempo local e ser capaz de tomar uma decisão, sem interferência do usuário, se deve irrigar o jardim ou não. Na situação em que a previsão do tempo indicar que irá chover em breve, o sistema deve entender que não há necessidade de irrigar o jardim, pois o solo será molhado em breve pela chuva. No caso que considera necessário irrigar, deve ser também capaz de realizar a irrigação.

Além do objetivo principal, também é meta do trabalho fazer com que esse sistema seja capaz alterar o modo de operação, ao comando do usuário, e irrigar o jardim automaticamente, e também informar ao usuário sobre as irrigações realizadas.

1.2 Motivação

Atualmente as pessoas têm se preocupado mais com conforto e economia de recursos. Uma maneira de providenciá-los à população é por meio da automatização residencial.

> A presença da automação na economia global é crescente e ultrapassou as fronteiras das instalações industriais. O esforço diário de conjugação de dispositivos automáticos com ferramentas organizacionais e matemáticas tem levado à criação de sistemas complexos aplicáveis às várias atividades humanas. Assim, não somente a manufatura e processos industriais vêm sendo automatizados, como também os serviços de infra-estrutura, os escritórios e, até mesmo, os lares.(GUTIERREZ; PAN, 2008)

Um projeto de irrigação de jardins autônomo é um projeto de engenharia que busca a otimização dos recursos hídricos de uma casa disponíveis para o jardim e, com isso, maior conforto para os residentes da casa.

Além disso, o agronegócio é um setor muito forte na economia brasileira. Segundo dados do Cepea-Esalq/USP (Centro de Estudos Avançados em Economia Aplicada), o agronegócio correspondeu a $21,4\%$ do PIB (Produto Interno Bruto) do Brasil no ano de 2019[13], porcentagem equivalente a R\$$1,56$ trilhões.[14] No primeiro trimestre de 2020, houve um crescimento de $3,3\%$[15]. Mais precisamente, o ramo agrícola cresceu $1,91\%$ nesse período.

A agropecuária é o setor da economia que mais consome água. De acordo com a *Conjuntura dos recursos hídricos no Brasil* (2019), no ano de 2018 somente a irrigação correspondeu a $66,1\%$ do total de água consumida no Brasil, o que equivale a um consumo de $1.101 m^3/s$ de água. O relatório também indica que a previsão de demanda é crescente ao longo dos anos, podendo aumentar 26% até a próxima década. Também segundo a Agência Nacional de Águas (ANA), quase metade da água utilizada na agropecuária é desperdiçada por diversos motivos, como irrigações mal executadas e falta de controle da quantidade de água utilizada.

Sendo assim, o projeto descrito e desenvolvido tem um potencial econômico muito forte, pois pode ser adaptado e expandido para modernizar e gerar economia de água, energia e recursos em grande escala.

---

[13] Dados disponíveis em: <https://www.cnabrasil.org.br/assets/arquivos/boletins/sut.pib_dez_2020.5mar2020vf.pdf>. Acesso em 17 jun. 2020.

[14] Fonte: IBGE <https://www.ibge.gov.br/explica/pib.php>. Acesso em 20 jun. 2020.

[15] Dados disponíveis em <https://www.cepea.esalq.usp.br/upload/kceditor/files/Cepea_PIB_Agro_marco_junho2020(2).pdf>. Acesso em 17 jun. 2020.

1.3 Organização do texto

Esta monografia está dividida em 6 capítulos. O Capítulo 1 introduz o problema tratado no trabalho, bem como sua contextualização, motivação e objetivos.

O Capítulo 2 apresenta a revisão bibliográfica, que engloba toda a base teórica que estrutura o projeto de irrigação proposto. Neste capítulo, são apresentados o conceito de Internet das Coisas, a arquitetura cliente-servidor utilizada em redes de computadores, o protocolo MQTT usado em aplicações de Internet das Coisas, e o protocolo TCP/IP.

A descrição do sistema é apresentado no Capítulo 3, onde são levantados os requisitos de projeto e desenvolvidos os projetos de *hardware*, que engloba a definição dos componentes físicos e o esquemático eletrônico, e de *software*, que descreve a modelagem do comportamento lógico do sistema.

O Capítulo 4 trata da implementação da modelagem do *software* previamente descrita no capítulo anterior, e também alguns testes de validação individuais realizados. Essa implementação é dividida pelas frentes implementadas.

Em seguida, o Capítulo 5 apresenta os resultados obtidos e algumas discussões sobre o projeto ao todo.

Por fim, o Capítulo 6 apresenta as conclusões do projeto, bem como as propostas de trabalhos futuros levando como base o presente trabalho.

2 Revisão Bibliográfica

Com o objetivo de entender a fundamentação teórica por trás da Internet das Coisas, foi feito, primeiramente, um estudo sobre seu conceito e aplicações.

Em seguida, foi realizada uma revisão bibliográfica referente à transmissão de mensagens pela internet, que engloba a arquitetura cliente-servidor e os protocolos utilizados durante o desenvolvimento deste projeto.

São eles: protocolo MQTT (*Message Queuing Telemetry Transport*), um dos protocolos mais populares para aplicações de Internet das Coisas; protocolo TCP/IP (*Transmission Control Protocol/Internet Protocol*), padrão mais aceito para envio de mensagens pela internet; protocolo HTTP, utilizado para sistemas de hipermídia; e protocolo NTP (*Network Time Protocol*), usado para sincronizar relógios na internet.

2.1 Internet das Coisas

O termo Internet das Coisas (ou *Internet of Things*, *IoT*, em inglês) foi criado em 1999 por Kevin Ashton. Segundo Ashton (2009), sua intenção era dizer que o mundo real depende de coisas físicas, e não apenas ideias e informações e, por isso, o termo surgiu com o intuito de juntar as informações sobre as coisas do mundo real na internet. Para o especialista, a partir do momento em que computadores conseguem reunir informações dessas coisas ou objetos sem ajuda humana, é possível otimizar diversas atividades e reduzir perdas e custos. De acordo com o Oxford University Press (OUP) (2019), o conceito de Internet das Coisas é usado para se referir à interconexão, via internet, de sistemas embarcados presentes no dia-a-dia, possibilitando que estes recebam e transmitam informações.

A partir do compartilhamento das informações, a Internet das Coisas origina uma variedade de novos recursos com dados armazenáveis, que podem ser acessados remotamente e utilizados nas mais diversas aplicações para a sociedade, por exemplo nas áreas de saúde, varejo, segurança ou ambientes inteligentes.

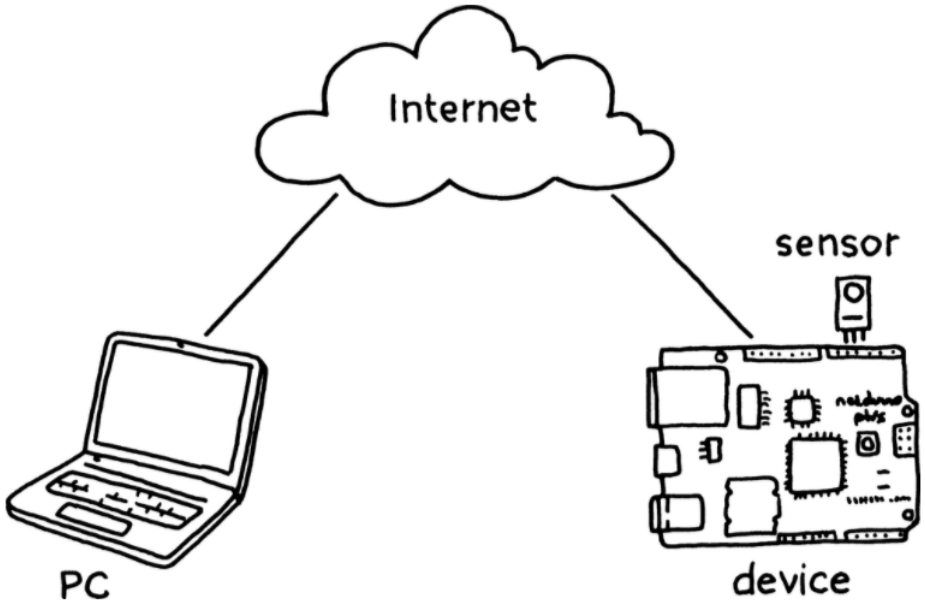
Atzori, Iera e Morabito (2010) citam, na área da saúde, o uso de sensores sem fio que permitem monitorar continuamente o estado do paciente, mesmo à distância. Para economizar energia é possível conectar os dispositivos da rede elétrica, que compartilham informações em tempo real e otimizam a distribuição de energia.

No varejo, cita que a aplicação de Internet das Coisas permite monitorar todo o processo comercial, desde a fabricação do produto até a logística de transporte e distribuição, e o nível de produtos disponíveis em estoque.

Os autores também mencionam ambientes inteligentes, como cidades e casas, onde as tecnologias são aplicadas para melhorar a qualidade de vida urbana. Nas cidades, por exemplo, é possível monitorar o estado e desempenho de calçadas, ruas e corredores de ônibus. A distribuição de energia também pode ser otimizada de acordo com a demanda. Nas casas inteligentes é possível que o morador gerencie eletrodomésticos e ambientes, controlando luminosidade e temperatura.

A ideia da Internet das Coisas consiste em uma maneira única de mapear objetos comuns a representações virtuais, similar à estrutura da internet, uma rede interconectada de computadores. Esses objetos podem coletar informações sobre si mesmos ou transmitir dados obtidos a partir de sensores. A Figura 2.1 expressa essa relação. A partir da premissa de que os objetos possam ser unicamente endereçados e tenham a capacidade de se conectar à internet, a informação coletada por eles pode fluir da mesma maneira que dados fluem de computadores à internet, utilizando os mesmos protocolos (AGGARWAL; ASHISH; SHETH, 2013).

Figura 2.1 – Computador e dispositivo IoT conectados pela internet



Fonte: extraído de Pfister (2011).

2.2 Arquitetura cliente-servidor

A internet como é conhecida atualmente é baseada em uma arquitetura de comunicação chamada cliente-servidor. Essa comunicação pode ser feita por meio de uma rede de computadores ou por um único computador.

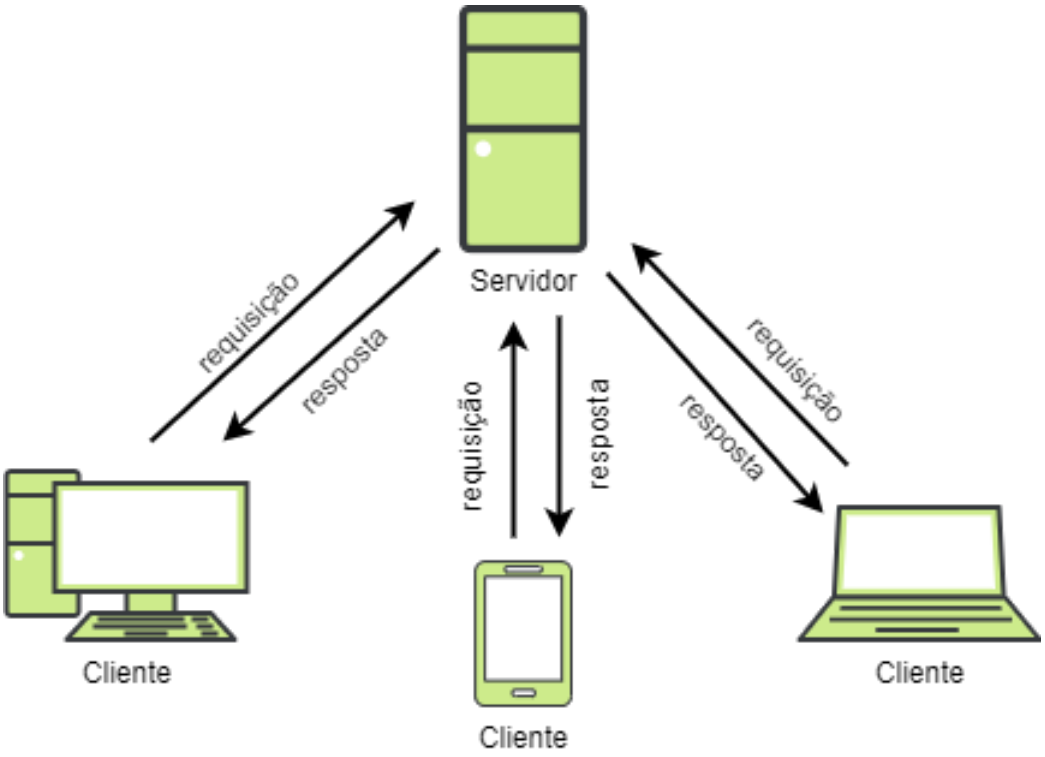
Nesse modelo, o servidor é um *software* que está sempre em execução, esperando uma requisição do cliente. Ele é responsável por armazenar ou processar dados ou serviços

de uma aplicação, e deve estar sempre disponível ao cliente, desde que este saiba sua localização. Ao receber um pedido, o servidor deve processá-lo e responder fornecendo os dados ou serviços solicitados ao cliente.

O cliente é um *software* que inicia a comunicação (conexão) com o servidor para fazer uma solicitação de dados e espera por uma resposta, utilizando a rede de computadores para a comunicação. Geralmente o cliente é acionado por um usuário, mas pode também ser acionado por uma rotina automática (OLIVEIRA, 2017).

A Figura 2.2 mostra a conexão básica entre computadores em uma arquitetura cliente-servidor para um único servidor. Um cliente pode fazer requisições a diversos servidores, assim como um servidor pode responder a diversos clientes. Isso permite uma descentralização dos serviços e dados, que estão distribuídos em servidores pelo mundo todo. Desse modo, a funcionalidade geral de um serviço fica disponível para todos os clientes, mas não precisa ser implementada por todos os serviços (SOMMERVILLE, 2013).

Figura 2.2 – Estrutura de uma arquitetura cliente-servidor.



Fonte: autoria própria.

Um dispositivo de Internet das Coisas pode atuar tanto como cliente quanto como servidor. No papel de cliente, pode-se citar um dispositivo que faz uma solicitação da previsão do tempo para um servidor externo, ou um dispositivo que faz a leitura de temperatura do ambiente e comunica a leitura a um banco de dados. Na função de servidor, o dispositivo pode atuar com as funções mencionadas, como servidor *web* ou servidor de banco de dados.

Para funcionar, o servidor deve ter instalado um sistema operacional, como *Linux* ou *Windows*, que reconheça a rede cliente-servidor. A comunicação entre cliente e servidor é feita por meio de protocolos. Na arquitetura cliente-servidor é importante que todas as aplicações consigam se conectar e comunicar pela rede. Para isso, é necessário que utilizem o mesmo protocolo de comunicação.

2.3 Protocolo MQTT

O protocolo MQTT (*Message Queuing Telemetry Transport*) é um protocolo de troca de mensagens assíncronas entre máquinas que segue a arquitetura cliente-servidor. Ele foi criado em 1999 por Andy Stanford-Clark (*IBM*) e Arlen Nipper (*Eurotech*)[1]. Esse protocolo é bastante popular em aplicações de Internet das Coisas, pois possui vantagens como código aberto, flexibilidade, padronização, possui estrutura simples e é um protocolo leve. Isso permite que seja aplicado em dispositivos e sistemas restritos e em redes com baixa largura de banda e com alta latência.

Segundo Oliveira (2017), o MQTT utiliza um *broker*, ou seja, um *software* intermediador que atua como servidor, que tem função de gerenciamento de mensagens, tanto recebendo quanto repassando mensagens, quando solicitado. Sendo assim, atua nos dois sentidos, de recebimento e transmissão de dados. O *broker* deve implementar um armazenamento de dados estruturado por tópicos.

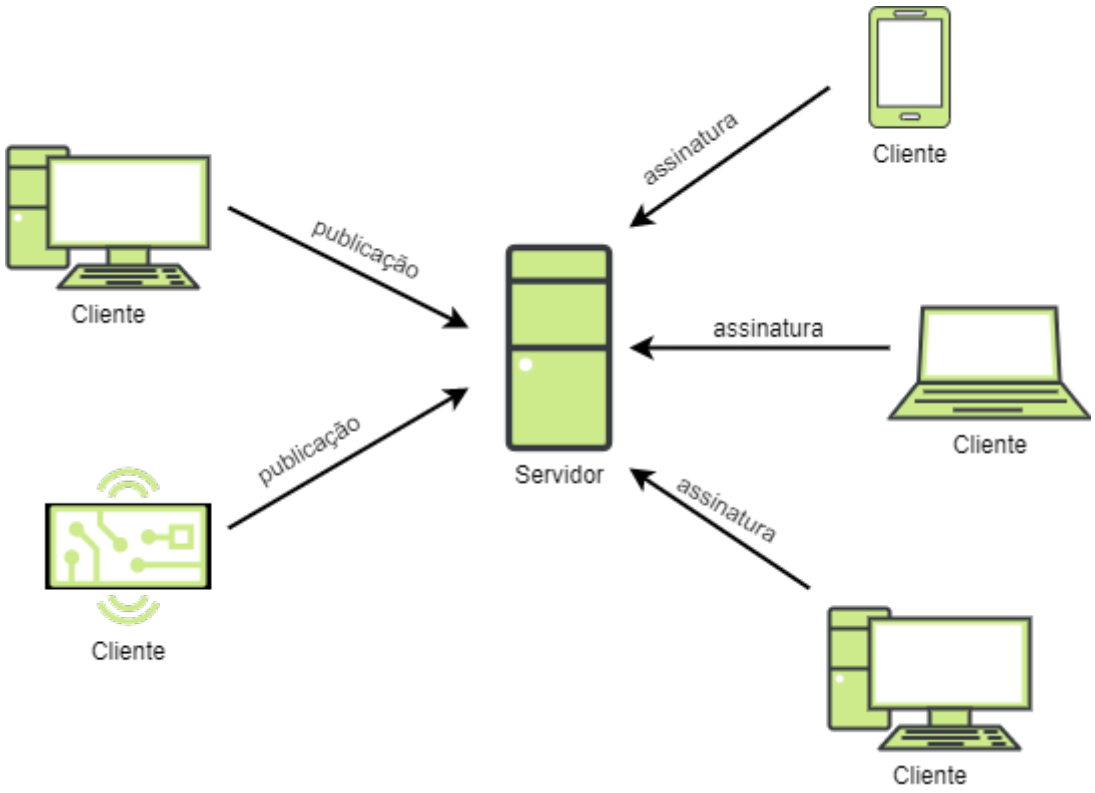
Por outro lado, "um cliente é qualquer coisa que possa interagir com o *broker* e receber mensagens"(YUAN, 2017). Para aplicação em Internet das Coisas, os sensores geralmente fazem o papel de clientes que se conectam ao servidor para publicar mensagens. Outros clientes, como aplicativos que processam dados, podem também se conectar ao servidor para receber essas mensagens (MARTINS; ZEM, 2015). Dessa maneira, os clientes podem ser tanto a origem quanto o destino das mensagens. Esse é o modelo de publicação-assinatura (*publish-subscribe*, em inglês).

Para Yuan (2017), tal modelo funciona da seguinte maneira: primeiramente, um cliente se conecta ao *broker* e pode assinar um tópico de mensagens. Essa conexão cliente-servidor é feita geralmente por meio do protocolo TCP/IP. Então, outro cliente conecta-se ao *broker* e publica uma mensagem em determinado tópico. A seguir, o *broker* é responsável por encaminhar tal mensagem a todos os clientes assinates do tópico em questão. A Figura 2.3 esquematiza essa troca de informações:

O MQTT está atualmente na versão 5.0 e já é classificado como padrão *OASIS* (*Organization for the Advancement of Structured Information Standards*, ou Organização para o Avanço de Padrões em Informação Estruturada em português, que é uma organização

[1] Fonte: <http://mqtt.org/faq>.Acesso em 18 jun. 2020.

Figura 2.3 – Estrutura padrão do MQTT



Fonte: autoria própria.

global sem fins lucatrivos que tem como objetivo legalizar e padronizar serviços *web*.[2])

De acordo com a documentação do MQTT v5.0[3], existem 16 tipos de mensagens que podem ser transmitidas pelo protocolo. A Tabela 2.1 apresenta essas categorias. Os principais tipos são: *CONNECT*, *PUBLISH* e *SUBSCRIBE*. A Figura 2.4 é um diagrama de sequência que mostra um exemplo de fluxo de mensagens que pode ocorrer entre dois clientes e um servidor.

Para que uma mensagem possa ser enviada via protocolo MQTT, é necessário definir o IP do *broker*, o tópico da mensagem e também algumas configurações de conexão entre cliente e servidor (como usuário e senha do MQTT, e nome da rede e senha do Wi-Fi ao qual será feita a conexão), além da mensagem em si que será publicada:

- **IP do *broker***: informação necessária para que a mensagem consiga chegar ao servidor de destino correto. Tanto o cliente remetente quanto o cliente destinatário precisam ter acesso ao número do IP para poderem se conectar ao servidor.
- **Tópico da mensagem**: é a identificação da categoria da informação publicada. O tópico é usado para enviar as mensagens aos respectivos assinantes, e é composto por uma *string*, ou seja, uma sequência de caracteres.

[2] Fonte: <https://www.oasis-open.org/org>. Acesso em 20 jun. 2020.
[3] Fonte: <http://mqtt.org/documentation>. Acesso em 19 jun. 2020.

Tabela 2.1 – Tipos de mensagem do protocolo MQTT

Nome	**Valor**	**Fluxo**	**Descrição**
Reserved	0	Proibido	Reservado
CONNECT	1	Cliente → Servidor	Requisição para conectar ao servidor
CONNACK	2	Servidor → Cliente	Reconhecimento da conexão
PUBLISH	3	Cliente ⟺ Servidor	Publicação de mensagem
PUBACK	4	Cliente ⟺ Servidor	Reconhecimento da publicação
PUBREC	5	Cliente ⟺ Servidor	Publicação recebida (garantia de entrega parte 1)
PUBREL	6	Cliente ⟺ Servidor	Publicação liberada (garantia de entrega parte 2)
PUBCOMP	7	Cliente ⟺ Servidor	Publicação completa (garantia de entrega parte 3)
SUBSCRIBE	8	Cliente → Servidor	Requisição de assinatura do cliente
SUBACK	9	Servidor → Cliente	Reconhecimento de assinatura
UNSUBSCRIBE	10	Cliente → Servidor	Requisição de cancelamento da assinatura
UNSUBACK	11	Servidor → Cliente	Reconhecimento do cancelamento da assinatura
PINGREQ	12	Cliente → Servidor	Requisição PING
PINGRESP	13	Servidor → Cliente	Resposta PING
DISCONNECT	14	Cliente → Servidor	Desconexão do cliente
AUTH	15	Cliente ⟺ Servidor	Troca de autenticação

Fonte: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html>. Acesso em 19 jun. 2020.

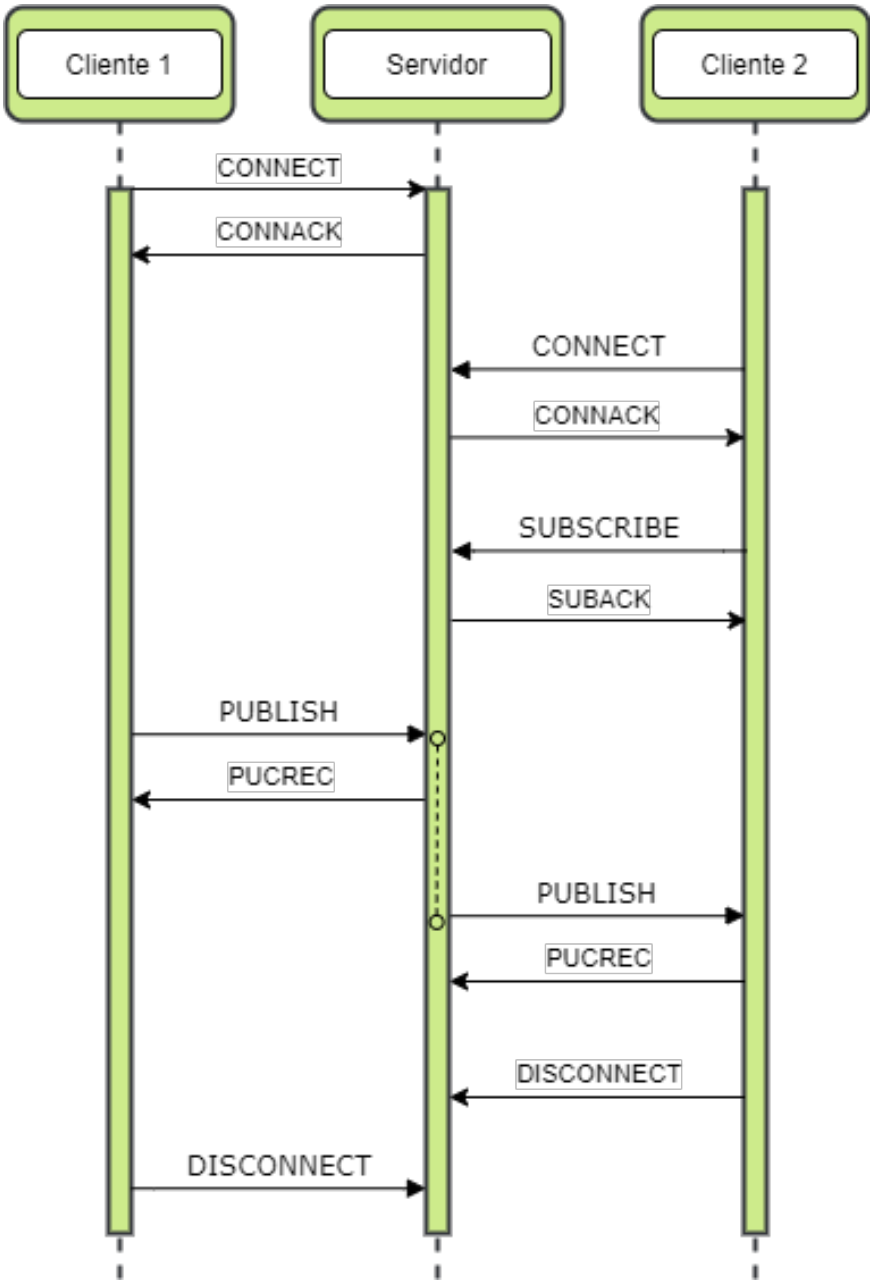
- **Nome do usuário e senha do MQTT**: são valores definidos no arquivo de configuração do servidor. Essas configurações também devem ser conhecidas pelos clientes, para obterem acesso ao servidor. Essas configurações não são obrigatórias, mas é recomendável utilizá-la por questões de segurança.

- **Nome da rede Wi-Fi**: também chamado de SSID (*Service Set Identifier*), é o nome de identificação associado a uma rede Wi-Fi, que se encontra disponível para o usuário identificar essa rede.

- **Senha do Wi-Fi**: é a senha de acesso ao Wi-Fi, necessária para que os dispositivos obtenham acesso à rede.

O protocolo MQTT foi desenvolvido com base no protocolo TCP/IP para fazer a transmissão de dados, pois necessita de conexões ordenadas, sem perdas e bidirecionais. O MQTT especifica como são organizados os *bytes* de dados para transmissão por meio da rede, usando o TCP/IP (YUAN, 2017).

Figura 2.4 – Exemplo de fluxo de mensagens usando protocolo MQTT



Fonte: adaptado de Khusnutdinov et al. (2018).

2.4 Protocolo TCP/IP

O protocolo TCP/IP é um conjunto de protocolos que é modelado na estrutura de camadas e é baseado na arquitetura cliente-servidor. Seu nome tem origem em dois protocolos: TCP, ou seja, *Transmission Control Protocol* (em português, Protocolo de Controle de Transmissão), e IP, ou seja *Internet Protocol* (em português, Protocolo da Internet) (PARZIALE et al., 2006).

Esse protocolo possui código aberto e tem o objetivo de construir uma interconexão entre redes. Sua vantagem é permitir a comunicação entre componentes separadas por

uma grande distância física. De acordo com Parziale et al. (2006), para que possa existir uma conexão entre duas redes, é necessário haver um computador associado a essas redes, que seja capaz de encaminhar pacotes de dados entre uma rede e outra. Esse é o chamado roteador.

O TCP/IP é modelado em uma pilha de camadas. Isso permite que haja uma divisão de tarefas e a possibilidade de desenvolver implementações alternativas de camadas. A comunicação entre elas é feita por meio de interfaces. Na estrutura de pilha, uma camada fornece um serviço à camada diretamente acima dela, e utiliza os serviços da camada abaixo (PARZIALE et al., 2006).

As camadas são apresentadas na Tabela 2.2 e descritas a seguir:

Tabela 2.2 – Camadas do protocolo TCP/IP

Aplicações
Transporte
Rede de internet
Interface da rede e *hardware*

Fonte: Parziale et al. (2006)

- Camada de aplicações: essa camada é fornecida pela aplicação que utiliza o protocolo para comunicação. Uma aplicação é definida como um processo de usuário que age em conjunto com outro processo, geralmente em um servidor diferente.

- Camada de transporte: essa camada fornece transferência de dados de uma ponta a outra. Ela atua entregando dados de uma aplicação ao seu destino remoto. Essa camada utiliza o TCP, que dá nome ao protocolo. O TCP fornece conexão orientada, entrega confiável, supressão de dados duplicados, controle de congestionamento e controle de fluxo.

- Camada de internet: essa camada fornece uma imagem de rede virtual usando o IP, que, por sua vez, fornece uma função de roteador. Ele é um protocolo sem conexão e não pode assumir que os dados que vieram abaixo dele são confiáveis.

- Camada de interface e *hardware*: essa camada é a interface ao *hardware*. Pode ser orientada a pacotes ou a transmissões. Não há um protocolo padrão para tratar dessa camada.

O TCP/IP se tornou o protocolo padrão principal para conexão de redes da internet, sendo utilizado com diversos propósitos, como aplicações multimídia e uso comercial.

2.5 Protocolo HTTP

O HTTP (*Hypertext Transfer Protocol*) é um protocolo da camada de aplicações do TCP/IP utilizado para obter um documento ou conjunto de documentos construídos em hipertexto, ou HTML (*HyperText Markup Language*). Segundo Beock et al. (2011), uma página construída em HTML, linguagem de marcação de hipertexto utilizada em páginas *Web*, é interpretada por um navegador, e este é considerado o cliente da dupla cliente-servidor.

O HTTP define a estrutura das mensagens trocadas por este protocolo. De acordo com MDN Web Docs (2017), essas mensagens podem ser requisições ou respostas, que possuem estruturas diferentes.

As mensagens de requisição possuem os elementos (MDN Web Docs, 2017):

- Método HTTP: é constituído por um verbo ou substantivo que indica a operação que o cliente requisitou. Alguns exemplos de utilização de métodos são os verbos *GET*, usado para obter um recurso da rede, e *POST*, usado para enviar dados através de um formulário.
- Caminho do recurso, ou seja, o endereço do objeto requisitado.
- Versão do protocolo HTTP. Atualmente, a versão mais recente é a versão HTTP 3.0.
- Cabeçalho, opcional, contendo informações adicionais.
- Corpo da mensagem, contendo o recurso requisitado, caso seja necessário.

Já as mensagens de resposta estão estruturadas pelos elementos:

- Versão do protocolo HTTP.
- Código de *status*, que serve para indicar a situação da requisição, ou seja, se ela foi bem sucedida ou não.
- Mensagem de *status*, descrição sobre o código de status.
- Cabeçalho HTTP, opcional.
- Corpo da mensagem com dados, também opcional.

2.6 Protocolo NTP

O Protocolo de Tempo para Redes (ou NTP - *Network Time Protocol*) é um protocolo utilizado para sincronizar um conjunto de relógios dos dispositivos de uma rede na ordem de nanossegundos, usando o princípio de cliente-servidor (MILLS, 1985).

Os servidores NTP formam uma rede hierárquica, em que a primeira camada corresponde a uma referência primária de tempo, como um relógio atômico. Cada camada fornece o tempo correto à camada seguinte[4].

De acordo com Júnior (2007), o objetivo do protocolo NTP é construir uma rede de sincronização entre relógios de máquinas conectadas à internet com o tempo real mundial. Um cliente NTP é inicialmente configurado com uma lista de pares que fornecem o tempo. Então, o protocolo determina o servidor de tempo mais adequado dentre os disponíveis na lista do cliente.

[4] Fonte: extraído de: <https://ntp.br/ntp.php>. Acesso em 08 nov. 2020.

3 Descrição do projeto

Neste capítulo, é feita a descrição e detalhamento do sistema de irrigação IoT desenvolvido.

Primeiramente, são apresentados os requisitos do projeto para que ele seja capaz de cumprir os objetivos do sistema. Para isso são apresentados os componentes selecionados para cumprir esses requisitos.

Em seguida, é feita a modelagem comportamental, estrutural e dinâmica do sistema, por meio de diagramas UML (*Unified Modeling Language*). O modelo comportamental descreve os casos de uso do sistema e como ele deve se comportar mediante cada um desses casos. O modelo estrutural define a organização do *software* em classes, e o modelo dinâmico representa as interações do sistema enfatizando o ordenamento cronológico das ocorrências.

3.1 Requisitos de projeto

A fim de projetar o sistema de irrigação é necessário primeiramente levantar os requisitos do projeto a partir dos objetivos do sistema, ou seja, as condições ou capacidades que o sistema deve possuir para cumprir os objetivos levantados.

1. O sistema deve ser capaz de tomar uma decisão sobre irrigar um jardim, baseada na umidade do solo e na previsão do tempo, e realizar automaticamente essa irrigação. Para isso, deve ser capaz de:
   - Coletar dados referentes à umidade do solo;
   - Coletar dados referentes à previsão do tempo na região;
   - Processar dados e decidir se deve haver irrigação;
   - Fornecer água ao jardim.
2. O sistema deve ser capaz de cumprir uma rotina periódica de irrigação, independente da umidade solo e da previsão do tempo. Para isso, é necessário:
   - Fornecer água ao jardim periodicamente.
3. O sistema deve ser capaz de exibir os dados referentes às irrigações realizadas ao usuário:
   - Registrar a quantidade de água utilizada;

- Registrar o dia e a hora;
- Possuir interface com o usuário.

Levantados os requisitos de projeto, os componentes necessários são selecionados, e deve-se então modelar os comportamentos desejados do sistema.

3.2 Sistema físico

Nesta seção são apresentados os componentes físicos utilizados, bem como suas características, funcionamento e montagem do sistema.

3.2.1 Componentes

No projeto foram utilizados sensores e atuadores no jardim, além de controladores para fazer o processamento das informações obtidas pelos sensores e *on-line*. Os componentes são descritos a seguir.

3.2.1.1 Sensor de umidade com módulo comparador

Esse sensor (Figura 3.1) detecta a variação de umidade no solo[1]. Ele possui uma saída digital que pode ser regulada por meio de um potenciômetro, e uma saída analógica, que faz leituras entre 1024 (solo seco) e 0 (umidade alta).

O princípio do funcionamento desse sensor é a variação da resistência de suas hastes (eletrodos) de acordo com a umidade do solo. A condutividade do solo é medida aplicando-se uma corrente nas hastes do sensor.

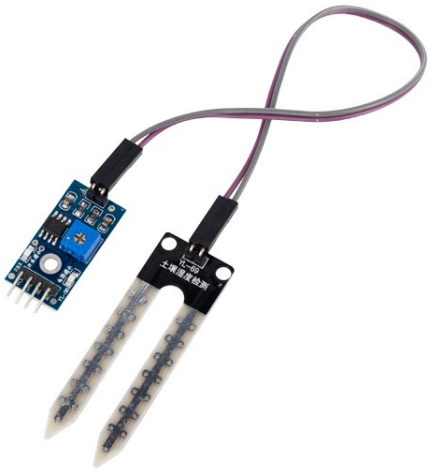
Quando ele está úmido, a condutividade é mais alta, devido à ocorrência de uma solução iônica no solo e, consequentemente, existe um maior fluxo de corrente entre os eletrodos. O sensor possui um circuito comparador conectado às hastes capaz de medir a condutividade.

Suas especificações são:

- Tensão de alimentação: 3,3 a 5V;
- Sensibilidade ajustável por potenciômetro;
- Saídas digital e analógica;
- Led indicador de detecção de umidade;
- Comparador: LM393.

[1] *Datasheet* disponível em: <https://www.curtocircuito.com.br/datasheet/sensor/umidade_do_solo.pdf>.

Figura 3.1 – Sensor de umidade do solo.



Fonte: extraído de: <https://www.filipeflop.com/produto/sensor-de-umidade-do-solo-higrometro/>

3.2.1.2 Válvula solenoide de vazão

Válvula normalmente fechada, usada para controlar o fluxo de água no jardim (Figura 3.2). A válvula é alimentada com 12V por uma fonte de alimentação conectada à rede elétrica, e acionada por um módulo relé.

O controle da válvula é feito por corrente elétrica, que passa por um solenoide, ou seja, uma bobina enrolada em forma de espiral. A energização dessa bobina cria um campo magnético que move um êmbolo e abre a válvula. Com a retirada de corrente elétrica do sistema, a válvula retorna ao seu estado fechado.

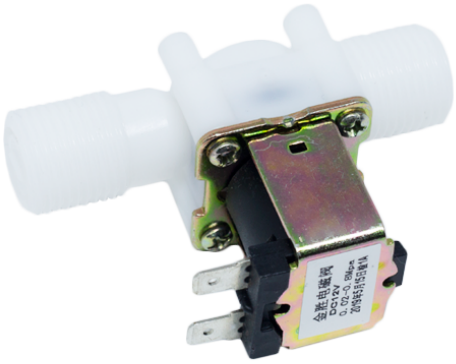
Suas especificações são:

- Tensão de operação: 12V;
- Corrente: 250mA;
- Modo de operação: normalmente fechada;
- Rosca de diâmetro 1/2”.

3.2.1.3 Módulo relé

O módulo relé (Figura 3.3) é utilizado para acionar a válvula solenoide. O relé funciona da seguinte maneira: ele possui uma bobina que, quando é alimentada com corrente elétrica, cria um campo magnético que atrai um contato e fecha ou abre um circuito.

Figura 3.2 – Válvula solenoide.



Fonte: extraído de: <https://www.filipeflop.com/produto/valvula-de-vazao-solenoide-agua-12vdc/>

A válvula é ligada ao contato normalmente aberto do relé. Esse contato se mantém aberto enquanto a bobina não está energizada, e fecha com a presença de corrente. Desse modo, o circuito da válvula se fecha apenas quando o pino do relé é energizado. O relé também possui um circuito normalmente fechado, com comportamento oposto, que se abre com a passagem de corrente elétrica.

- Tensão de operação: $3,3$V;
- Suporta carga de até: 100A;
- Corrente de operação: 20mA.

3.2.1.4 Sensor de fluxo

O sensor de fluxo (Figura 3.4) é utilizado para medir o fluxo de água que passa pela válvula[2]. Ele atua a partir de um sensor de efeito Hall, ou seja, que mede a diferença de potencial em um condutor elétrico.

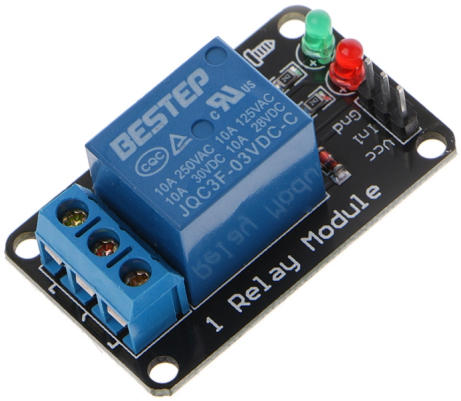
O sensor de fluxo possui esse sensor de efeito Hall fixo, e um ventilador com um ímã em seu eixo, de tal modo que, com o fluxo de água, a hélice gire, alterando a orientação do ímã. A rotação do ventilador gera então pulsos elétricos na saída do sinal do sensor.

Suas especificações são:

- Modelo: YF-S201B;

[2] *Datasheet* disponível em: <http://www.mantech.co.za/Datasheets/Products/YF-S201_SEA.pdf>.

Figura 3.3 – Módulo relé 3,3V.



Fonte: extraído de: <https://shopee.com.br/%E2%98%80S%E2%98%801PCS-M%C3%B3dulo-de-rel%C3%A9-de-canal-3V-de-3.3V-i.191704685.7804886212>

- Tipo de sensor: Efeito Hall;
- Fluxo de operação: 1-30 L/min;
- Tensão de operação: 4, 5-24V;
- Rosca de diâmetro 1/2".

Figura 3.4 – Sensor de fluxo de água.

Fonte: extraído de: <https://www.filipeflop.com/produto/sensor-de-fluxo-de-agua-12-yf-s201/>

3.2.1.5 Módulo ESP8266 NodeMCU-ESP12

Este módulo é uma placa de desenvolvimento e prototipagem que integra o controlador ESP8266-ESP12 (chip que possui Wi-Fi integrado), interface USB-Serial e regulador

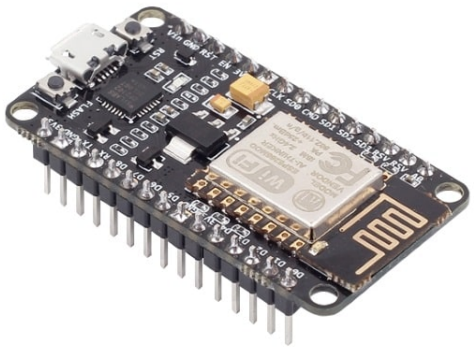
de tensão 3,3 V (Figura 3.5). Ele é muito usado em projetos de Internet das Coisas por causa da conexão sem fio e da ser facilmente programável pela IDE do Arduino.

O NodeMCU possui 11 pinos digitais e 1 pino analógico. Sua alimentação pode ser feita por USB ou por pino específico. A programação pode ser feita em linguagem LUA ou através da IDE do Arduino, que já possui diversas bibliotecas voltadas a essa placa.

Suas especificações são:

- Wireless padrão 802.11 b/g/n;
- Suporta 5 conexões TCP/IP;
- CPU de 32-bits;
- Tensão de operação: 3,3V;
- Tensão de alimentação: 4,5-9V;
- 11 pinos GPIO;
- 1 pino analógico com resolução de 10 bits.

Figura 3.5 – Placa de desenvolvimento NodeMCU ESP-12



Fonte: extraído de: <https://www.filipeflop.com/produto/modulo-wifi-esp8266-nodemcu-esp-12/>

3.2.1.6 Raspberry Pi

O Raspberry Pi é um mini computador de baixo custo desenvolvido no Reino Unido pela Fundação Raspberry Pi. Para utilizá-lo é necessário conectar um teclado, um mouse USB e um monitor a ele, além de um cartão de memória microSD (Figura 3.6). Neste projeto, é utilizado o Raspberry Pi 3 modelo B+.

No projeto, não está diretamente conectado ao restante dos componentes eletrônicos. Ele utiliza o sistema operacional Raspbian OS, distribuição Linux criada para rodar nos Raspberry Pi, e atua como um servidor (*broker*). Possui função de solicitar a previsão do tempo a uma API externa e fazer o processamento dos dados recebidos, tanto do sensor de umidade quanto da previsão do tempo. Ele também faz a comunicação com o módulo NodeMCU pela rede Wi-Fi.

Suas especificações são:

- Processador ARM Cortex-A53 Quad-Core;
- Memória RAM 1GB;
- Adaptador Wi-Fi integrado;

Figura 3.6 – Raspberry Pi 3 modelo B+.

Fonte: extraído de: <https://www.filipeflop.com/produto/raspberry-pi-3-model-b/>

Além dos componentes eletrônicos apresentados anteriormente, foram utilizados também componentes do subsistema hidráulicos para a montagem do sistema. Os componentes necessários foram mangueiras 1/2", adaptadores e encaixes para roscas da torneira, válvula solenoide e sensor de fluxo.

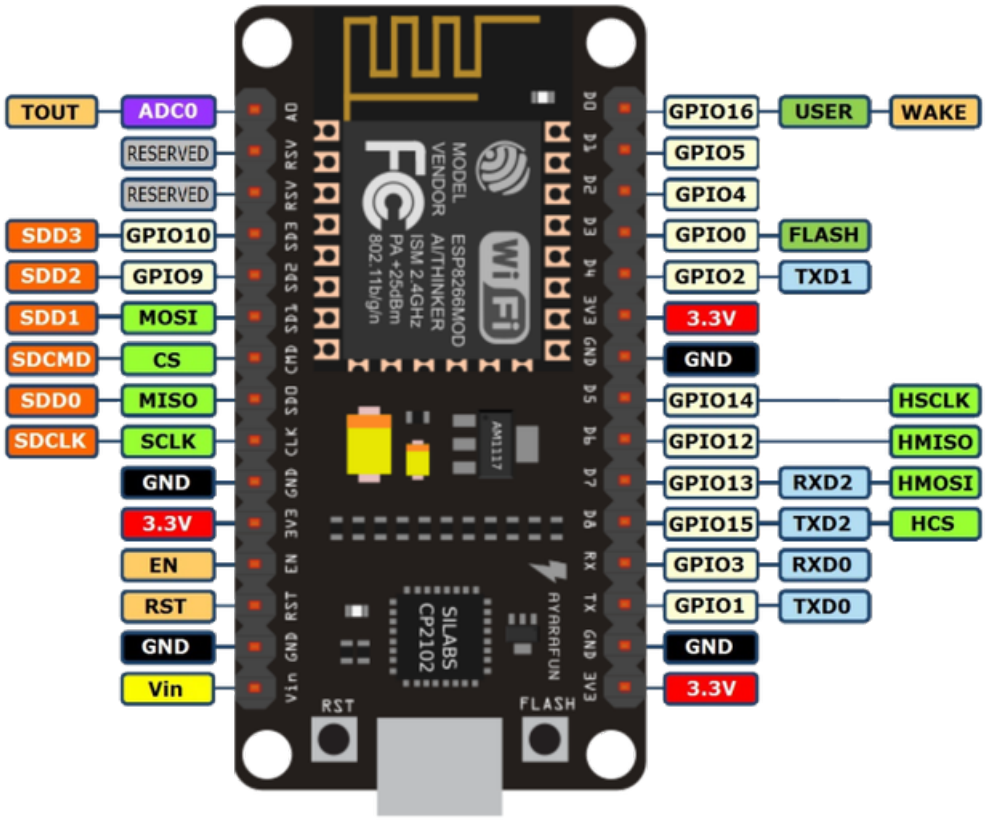
3.2.2 Esquemático

No que diz respeito ao subsistema eletrônico, o módulo NodeMCU atua como um controlador, pois deve ler os dados recebidos pelos sensores e também acionar a válvula solenoide através do módulo relé para liberação de água. O esquemático da montagem é mostrado na Figura 3.8.

O módulo ESP8266 (placa integrada ao NodeMCU) trabalha com regulador de tensão com nível de 3,3V. Dessa maneira, apenas o sensor de umidade e o módulo relé operam em uma faixa de tensão compatível com o controlador. Já o sensor de fluxo pode ser alimentado pelo pino VIN, único pino capaz de fornecer tensão de 5V, pois é conectado à alimentação da placa (interface USB), de acordo com o esquemático da placa. A pinagem do módulo NodeMCU está apresentada na Figura 3.7.

Desse modo, é necessário utilizar uma fonte de 12V para alimentar a válvula solenoide. Assim, quando acionado pelo controlador por um pino digital, o relé fecha o circuito entre a válvula e a fonte, permitindo sua abertura e passagem de água.

Figura 3.7 – Pinagem do NodeMCU.

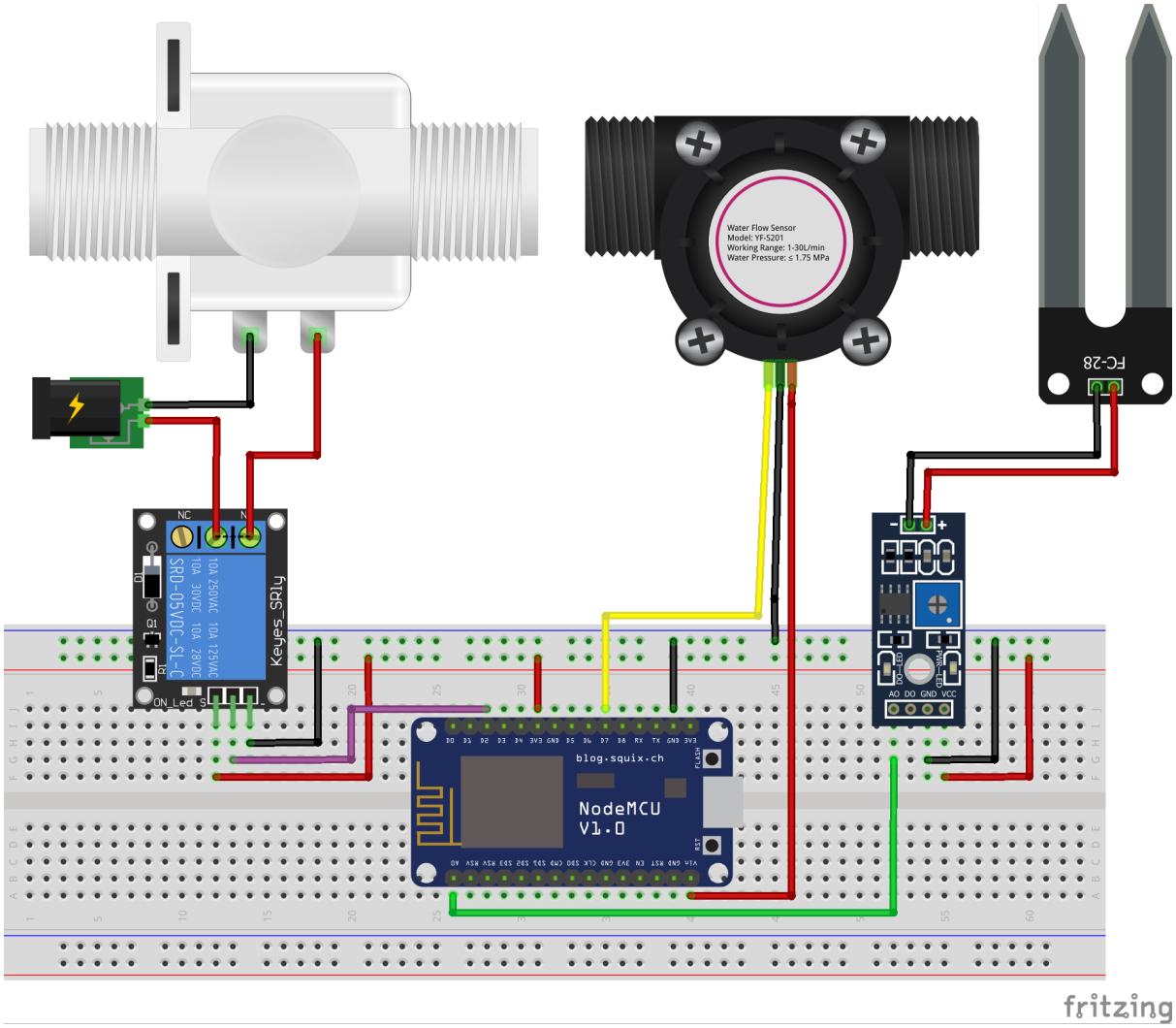


Fonte: extraído de: <https://blog.eletrogate.com/nodemcu-esp12-introducao-1/>.

As leituras dos sensores são feitas de maneiras distintas para cada sensor: o pino de saída do sensor de umidade é conectado ao pino analógico do módulo NodeMCU. Desse modo, o controlador consegue realizar leituras de umidade que variam entre 0 e 1024.

Já o pino de saída do sensor de fluxo é conectado a um dos pinos digitais de entrada, e o fluxo é calculado, tal como será explicado no capítulo de implementação, a partir da frequência de pulsos elétricos gerados nesse pino.

Figura 3.8 – Esquemático de montagem elétrica do projeto.



Fonte: autoria própria.

3.3 Modelagem comportamental

A modelagem comportamental do sistema descreve as funcionalidades que este deve cumprir. O projeto de *software* é feito partindo-se de diagramas UML. Eles descrevem, em detalhes, as tarefas designadas ao sistema e suas estruturas de dados, e também as relações com os atores que as realizam.

Os diagramas utilizados para modelar o comportamento foram: diagrama de casos de uso e diagrama de atividades. Esses diagramas descrevem o fluxo de atividades realizadas pelo sistema. Além destes, utilizou-se também o diagrama de classes para modelar a estrutura dos objetos que compõe o projeto de *software*.

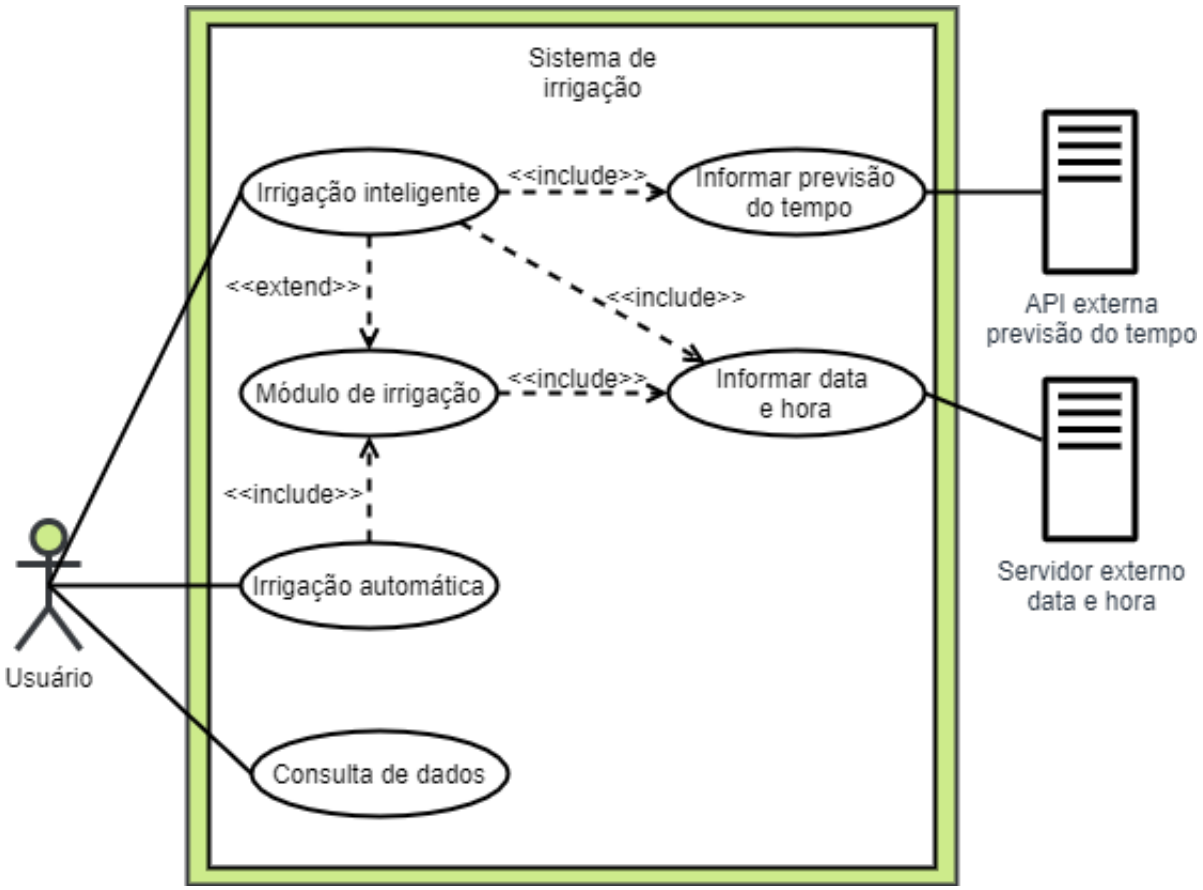
3.3.1 Diagrama de casos de uso

O diagrama de casos de uso tem como objetivo descrever cada cenário de utilização do sistema, ou seja, o que o sistema deve ser capaz de fazer, porém sem detalhamento em cada atividade em si Seidl et al. (2015). Esse diagrama deve descrever as relações entre os atores e os casos de uso do sistema.

As relações entre um caso de uso A e outro caso de uso B podem ser de extensão ou inclusão. Na relação de extensão, em que B é uma extensão de A, o caso de uso A pode utilizar o comportamento de B, porém não é necessário que isso ocorra. Essa situação é representada pela palavra-chave «extend» no diagrama. Já na relação de inclusão, em que A inclui B, o comportamento de B está integrado ao comportamento de A. Essa situação é representada pela palavra-chave «include» no diagrama (SEIDL et al., 2015).

Os casos de uso que possuem relação direta com um ator externo ao sistema identificados foram: irrigação inteligente, irrigação automática, módulo de irrigação, consulta de dados, informar previsão do tempo e informar data e hora. A Figura 3.9 apresenta esses casos.

Figura 3.9 – Diagrama de casos de uso.



Fonte: autoria própria.

1. **Irrigação inteligente**: é o caso de uso principal do sistema desenvolvido. Nele, o

sistema deve, uma vez ao dia, verificar a umidade do solo e a previsão do tempo e, com base nesses dados, tomar a decisão de irrigar o jardim ou não.

As pré condições que o sistema deve cumprir para esse caso de uso são: a conexão ao Wi-Fi e acesso à internet, para troca de mensagens, horário local e também obtenção da previsão do tempo; a capacidade de leitura do sensor de umidade; e um módulo de irrigação capaz de controlar o sistema hidráulico. A pós condição atingida por esse caso é: o jardim não necessita de água, podendo ter ocorrido irrigação ou não.

O ator desse caso é o usuário do sistema, e a ação que inicia o caso de uso é a solicitação do mesmo pelo modo inteligente.

2. **Irrigação automática**: nesse caso de uso, o solo é irrigado uma vez ao dia. As pré condições são o acesso ao Wi-Fi e à internet, para obtenção da data e hora, e também um módulo de irrigação capaz de controlar o sistema hidráulico. A pós condição é que o jardim estará irrigado.

   O ator desse caso também é o usuário do sistema, e o gatilho inicial é a solicitação do usuário pelo modo automático.

   Esse caso é desenvolvido tanto para efeito de testes e comparação com o caso anterior, com relação à água economizada, quanto para o caso de o usuário demandar de irrigação diária, independente da chuva. Isso pode ocorrer, por exemplo, em uma estufa que é fechada e não permite a entrada de águas pluviais.

3. **Módulo de irrigação**: esse caso de uso é uma sub-rotina dos casos de irrigação inteligente e automática. Na situação da irrigação automática, possui relação de inclusão, enquanto na irrigação inteligente possui relação de extensão.

   O módulo de irrigação consiste no ato de irrigação em si, ou seja, liberação de água para o jardim. Ele é acionado pelos outros casos de uso, não tendo o usuário controle direto sobre seu gatilho. Sua pós condição é ter o jardim irrigado.

4. **Consulta de dados**: nesse caso de uso, o usuário é capaz de consultar, por meio de interface gráfica, os dados da última irrigação. Esses dados consistem em: modo de irrigação, data e hora em que o sistema irrigou, volume de água utilizado na última irrigação e, caso a irrigação seja inteligente, a previsão do tempo obtida.

5. **Informar previsão do tempo**: já nesse caso, uma API externa de previsão do tempo é consultada pelo sistema. Esta retorna um objeto JSON (*JavaScript Object Notation*), formato de dados em linguagem computacional da forma "atributo e valor", com as informações climáticas da data e local desejados. A pré condição é a conexão à rede para troca de dados.

O ator aqui é o servidor externo que hospeda a API, que atua em resposta a uma solicitação do sistema desenvolvido nesse projeto. Esse caso de uso é uma inclusão da irrigação inteligente.

6. **Informar hora**: por fim, o último caso identificado diz respeito à obtenção da data e hora pelo sistema. Esse caso de uso possui relação de inclusão com ambos os casos de uso de irrigação, que fazem seu chamado. Seu objetivo é determinar se o sistema deve se manter em estado de espera ou iniciar as atividades de irrigação.

3.3.2 Diagrama de atividades

O diagrama de atividades modela em maiores detalhes cada caso de uso e as atividades realizadas durante cada um deles. Ele mostra a relação de ordem e dependência entre ações que compõe o caso de uso, e o fluxo de controle durante essas ações.

Esse diagrama é usado para descrever processos concorrentes que se comunicam, e pode ser comparado a uma Rede de Petri (SEIDL et al., 2015). O símbolo ⫛ indica que uma ação por si desencadeia outra atividade, que está hierarquicamente abaixo da atividade em questão.

1. **Irrigação inteligente**

   Nesse modo de operação de irrigação, o sistema inicialmente faz uma chamada ao servidor de horário local. Caso o horário obtido esteja entre determinada faixa, faz a leitura do sensor de umidade. Em seguida, envia os dados via protocolo MQTT ao Raspberry Pi, atuando como servidor e processador de dados. O Raspberry, então, solicita a previsão do tempo a uma API externa (*OpenWeather*[3]), para verificar se irá chover no dia atual ou no dia seguinte.

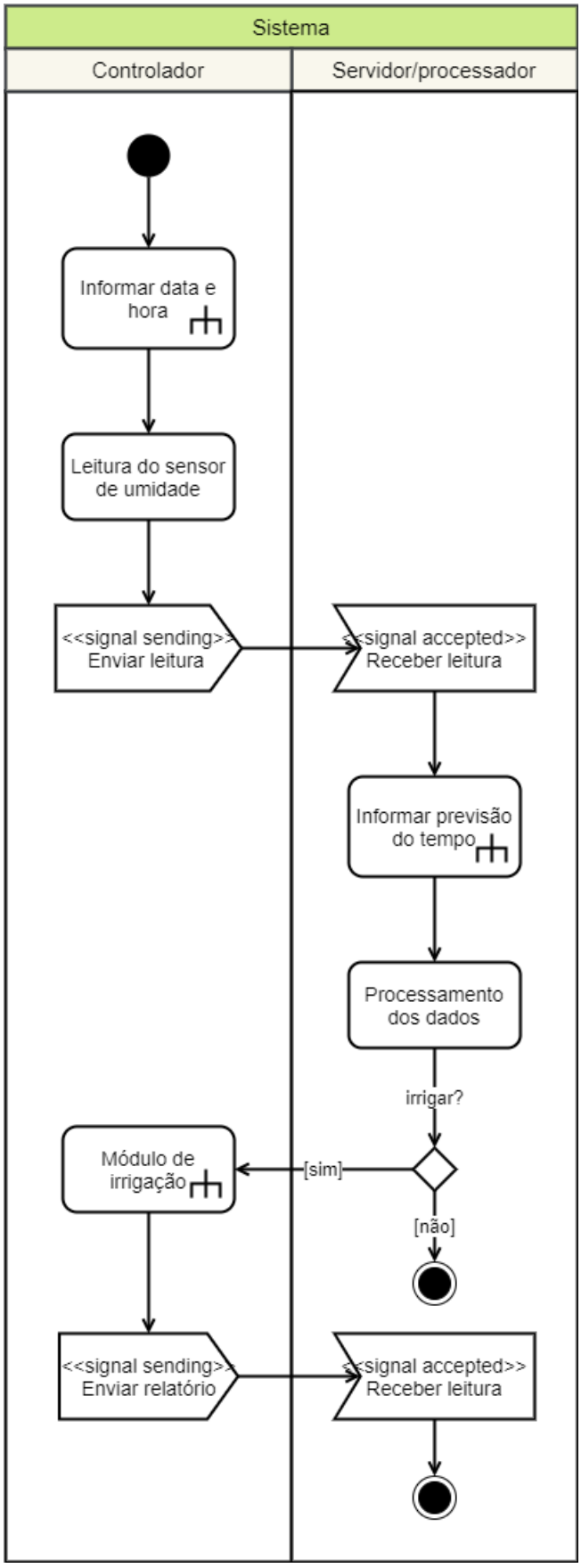
   Levando essas duas informações em consideração, o Raspberry então deve tomar a decisão de irrigar o jardim. Decidindo que sim, envia outra mensagem, também por meio de comunicação MQTT, para o controlador NodeMCU, que deve, então, acionar o módulo de irrigação. Ao final, o controlador envia os dados referentes à irrigação, como volume utilizado e horário da irrigação, ao servidor. O diagrama de atividades da irrigação inteligente é mostrado na Figura 3.10.

2. **Irrigação automática**

   Na irrigação automática, controlador obtém o horário local, também pela chamada ao servidor de horário. No horário determinado para irrigação, ativa o módulo de irrigação, como no caso anterior, porém sem necessidade de medição da umidade do solo nem consulta à previsão do tempo. A Figura 3.11 mostra o fluxo da atividade. Ao final, os dados referentes à irrigação também são enviados ao servidor.

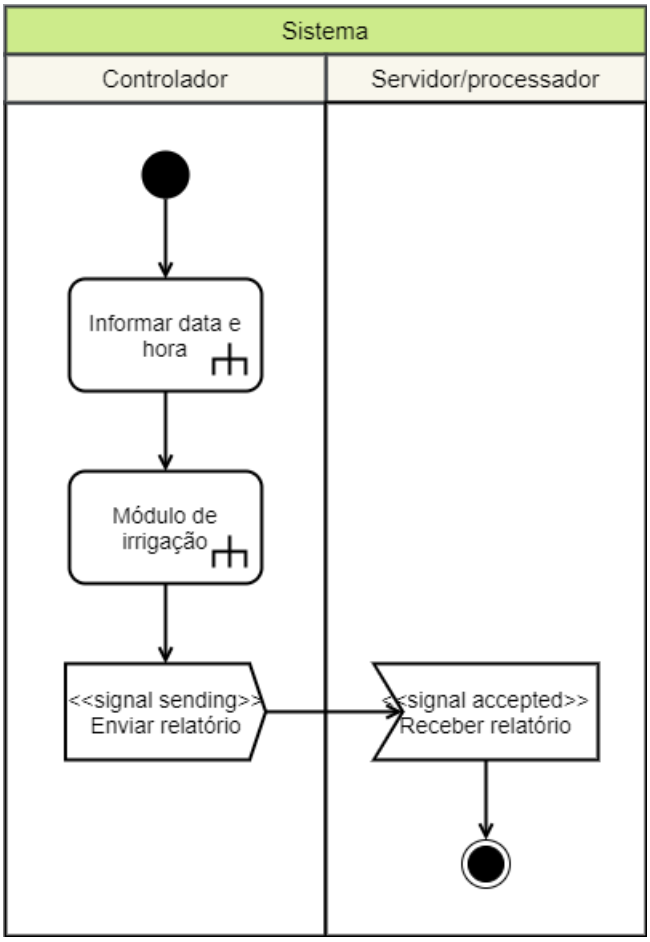
[3] Disponivel em: <https://openweathermap.org/>.

Figura 3.10 – Diagrama de atividades do caso de uso: Irrigação Inteligente.



Fonte: autoria própria.

Figura 3.11 – Diagrama de atividades do caso de uso: Irrigação Automática.



Fonte: autoria própria.

3. **Módulo de irrigação**

   O módulo de irrigação é uma sub-rotina dos casos de uso de irrigação inteligente e automática. Quando é chamado, o módulo deve abrir a válvula solenoide para liberar a passagem de água. A válvula se encontra conectada a um sensor de fluxo de água.

   Quando o sensor indica que o volume pré-determinado é atingido, o controlador deve fechar a válvula, cessando assim o fornecimento de água ao jardim. O diagrama representando essas atividades está representado pela Figura 3.12.

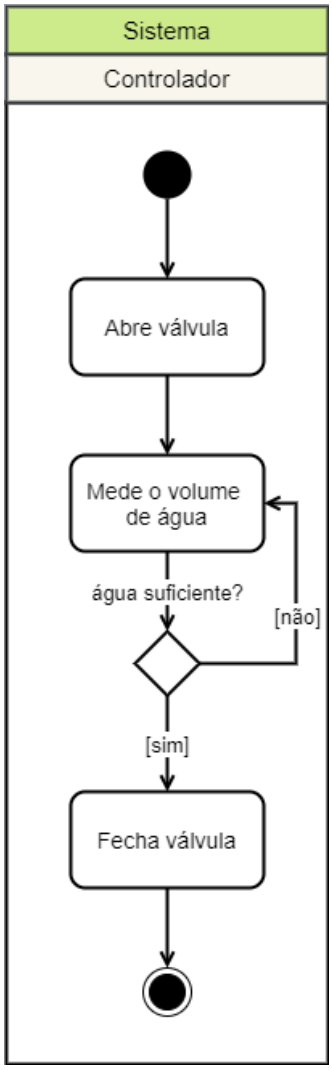
4. **Consulta de dados**

   Esse caso de uso diz respeito ao usuário que consulta os dados referentes à última irrigação realizada. Essa consulta é feita através de uma interface gráfica, hospedada localmente e servida pelo Raspberry Pi.

   Os dados levantados são: consumo de água, horário da irrigação, previsão do tempo (caso esteja no modo inteligente). A Figura 3.13 apresenta essas atividades.

5. **Informar previsão do tempo**

Figura 3.12 – Diagrama da atividade: Irrigação



Fonte: autoria própria.

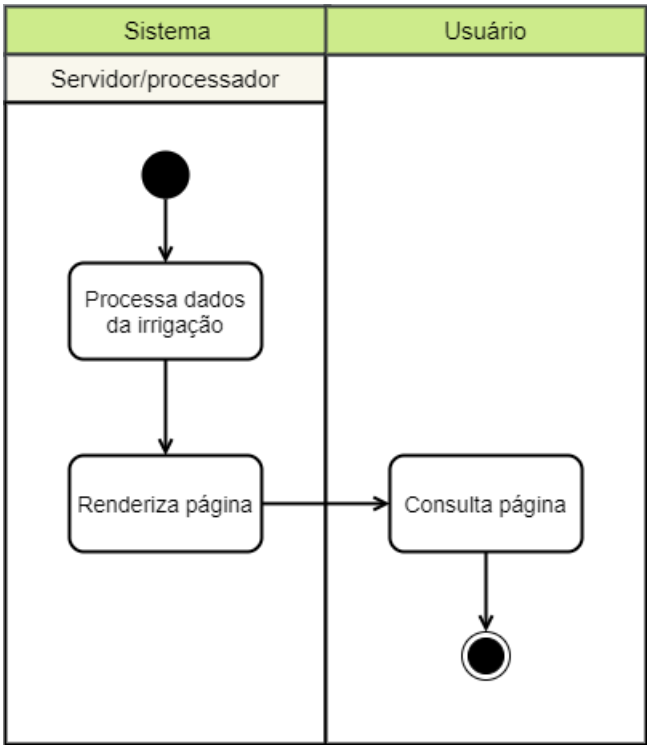
Nesse caso de uso, o Raspberry Pi faz uma consulta a uma API de servidor externo, *OpenWeather*. Essa consulta retorna um objeto em formato *JSON* contendo o tempo atual e a previsão para a semana seguinte. O sistema, porém, só necessita do tempo atual e do dia seguinte. O fluxo de mensagens está representado na Figura 3.14

6. **Informar hora**

O módulo NodeMCU, que atua como controlador, inicia-se em estado de espera. Ao "acordar", verificação que é feita periodicamente, faz uma requisição de um NTP) a um servidor externo que hospeda o Projeto NTP.br (<https://ntp.br/>). Esse projeto visa disponibilizar meios de sincronizar servidores de internet no Brasil.

Ao receber a hora local, o módulo verifica se a hora está entre um intervalo pré-configurado para continuar as atividades programadas (irrigação). Caso não esteja, retorna ao estado de espera. A Figura 3.15 mostra esse fluxo de informações.

Figura 3.13 – Diagrama de atividades do caso de uso: Consulta do consumo de água.



Fonte: autoria própria.

3.3.3 Modelagem estrutural

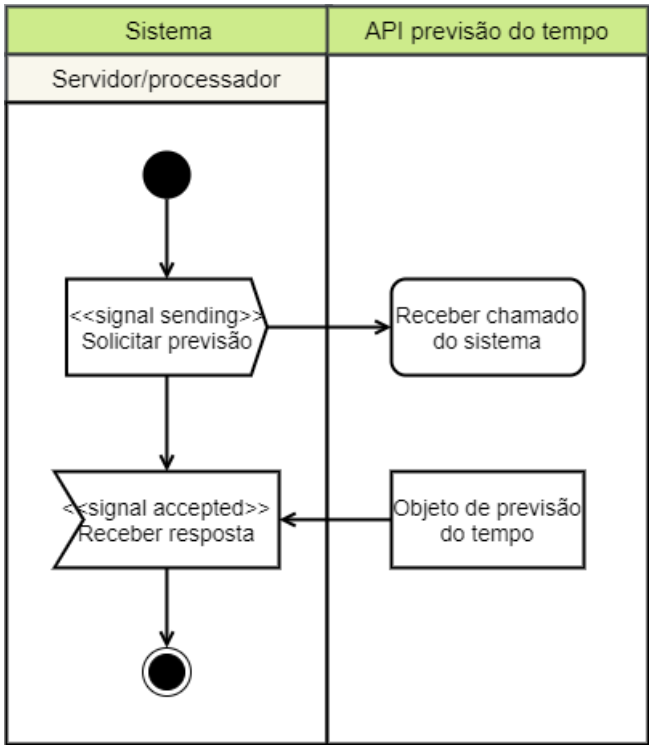
A modelagem estrutural é feita a partir de diagramas de classe. Esse tipo de diagrama é utilizado para modelar a estrutura de um sistema orientado a objetos. Uma classe é uma construção usada para representar um determinado conjunto de objetos similares. Um objeto é uma instância de uma classe.

Uma classe possui nome, atributos e operações. Os atributos são as características relevantes de uma instância da classe que possibilitam o armazenamento de informações, que podem ter diferentes valores entre os objetos, e as operações são as ações definidas para um objeto da classe, permitindo que estes se comuniquem e realizem ações e reações (SEIDL et al., 2015).

Uma interface é uma classe que implementa a interface gráfica do sistema. Ela possui a palavra-chave *«interface»* para sua diferenciação e pode ser usada por outras classes.

Já um diagrama de classes descreve as relações entre as classes existentes. No sistema modelado, foram identificadas duas classes e uma interface necessárias de implementação. As outras classes identificadas, referentes à comunicação, conexão Wi-Fi e protocolo MQTT, são utilizadas a partir de bibliotecas importadas. Elas não foram incluídas no diagrama visto que não é o escopo do projeto implementar essas bibliotecas, e serão melhor detalhadas no capítulo seguinte.

Figura 3.14 – Diagrama de atividades do caso de uso: Informar previsão do tempo.



Fonte: autoria própria.

As classes do sistema e relações entre si estão apresentadas no diagrama da Figura 3.16. São elas: *Jardim*, *Mensagem*, *Previsao* e *InterfaceWeb*.

1. Classe *Jardim*

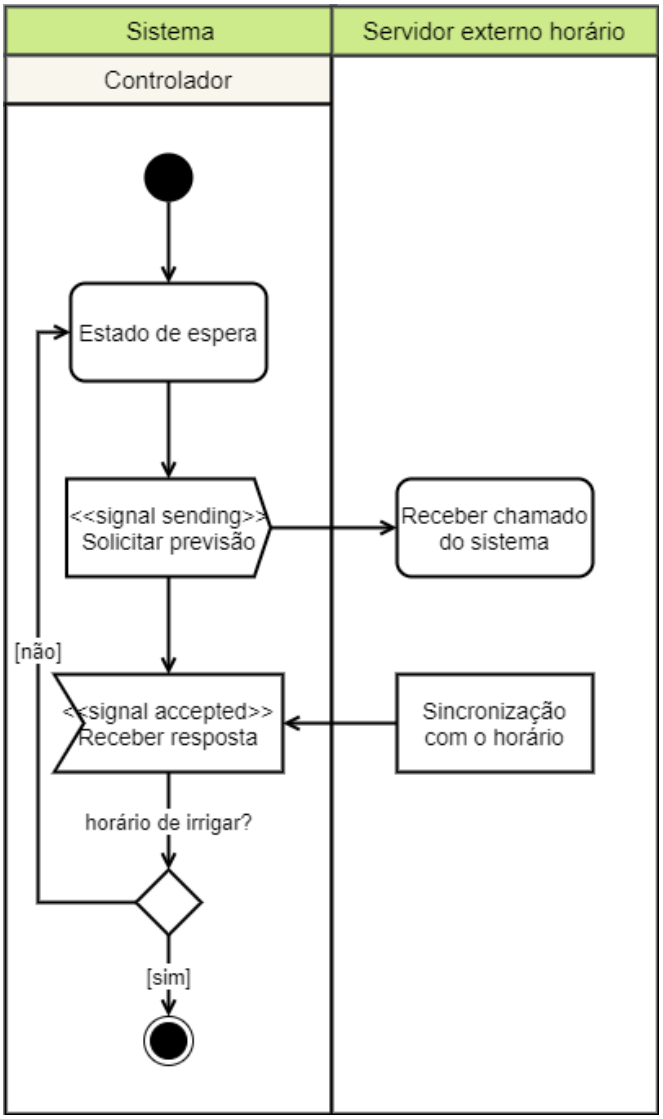
   Essa classe é implementada no módulo NodeMCU e modelada com o intuito de tratar dos componentes hidráulicos do sistema, como leitura de sensores e controle da válvula. Ela possui os seguintes atributos:

   - **contador**: variável utilizada para contabilizar os pulsos do sensor de fluxo;
   - **fluxo**: variável que armazena o fluxo de água passando pela mangueira, cujo cálculo é feito a partir da contagem de pulsos;
   - **volume**: variável destinada à medida do volume total de água usada;
   - **ultimo_check**: variável temporal correspondente à última verificação de horário que foi realizada pelo controlador.

   Já seus métodos são:

   - **controla_valvula**: método que controla a válvula de acordo com seu estado atual: abre se estiver fechada ou fecha se estiver aberta;
   - **calcula_fluxo**: é o método utilizado para fazer o cálculo do fluxo de água;

Figura 3.15 – Diagrama de atividades do caso de uso: Informar hora.



Fonte: autoria própria.

– **modulo_irrigacao**: método responsável por realizar todo o controle da irrigação, que engloba o controle da válvula por *feedback* do volume de água.

2. Classe *Mensagem*

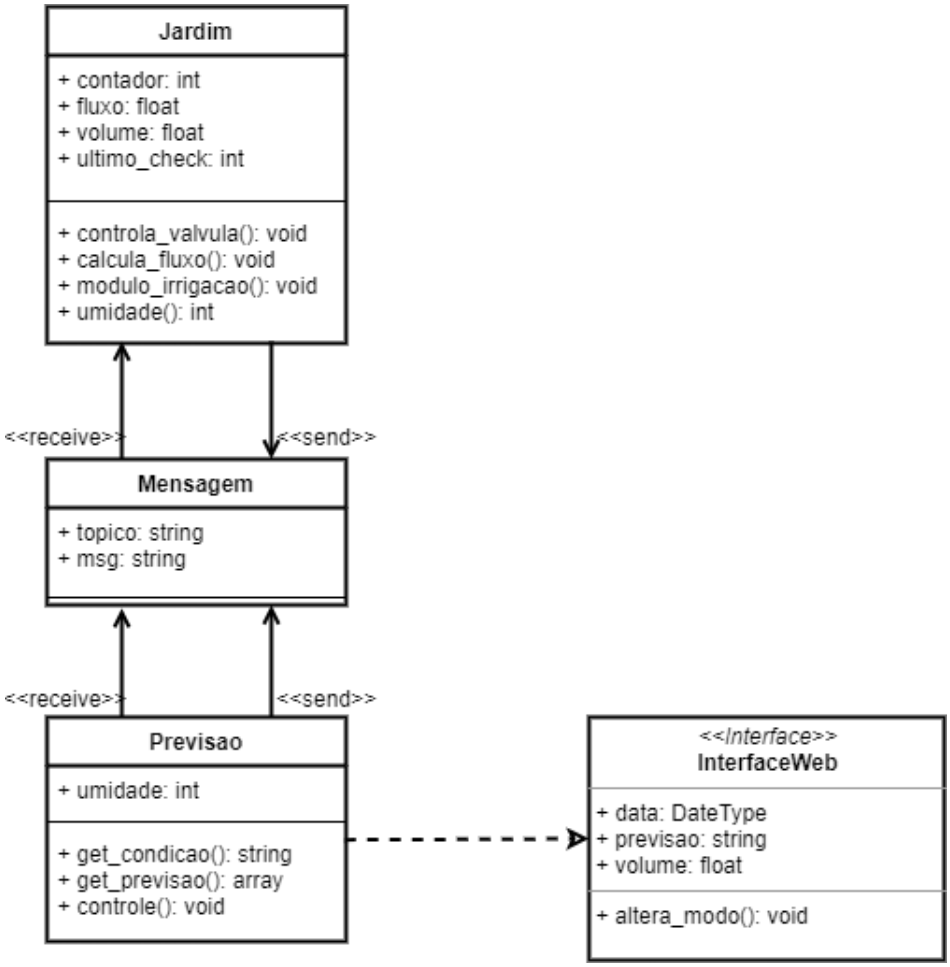
Essa classe representa um tipo de objeto que é a mensagem estruturada no formato necessário de tópico e mensagem para ser transmitida por meio do protocolo MQTT. Aqui não se considerou o IP do *broker* nem as configurações de segurança como atributos da classe, pois se entende que são constantes para o escopo deste sistema.

Essa classe não possui métodos, apenas dois atributos do tipo *string*. São eles:

– **topico**: refere-se ao tópico da mensagem, e

– **msg**: a mensagem em si que é enviada.

Figura 3.16 – Diagrama de classes.



Fonte: autoria própria.

3. Classe *Previsao*

   Essa classe é implementada no Raspberry Pi com o objetivo de processar os dados necessários para decidir se o sistema deve ou não irrigar o jardim, ou seja, a previsão do tempo e a umidade do solo. Ela possui apenas um atributo:

   – **umidade**: é o valor recebido do sensor de umidade, no formato *int* em porcentagem, usado no método de decisão.

   A classe também possui três operações:

   – **get_condicao**: método usado para processar os dados da API utilizada de previsão do tempo. Ele traduz os códigos de condição climática oriundos da API em texto, de acordo com a sua documentação.

   – **get_previsao**: operação que obtém a previsão do tempo por meio de uma requisição HTTP ao servidor da API externa. Além disso, essa operação faz o processamento dos dados obtidos para obter apenas os dados relevantes da requisição, que é a previsão do tempo do dia atual e do próximo dia em questão.

- **controle**: é o método de decisão, responsável por processar a umidade do solo e a previsão do tempo e decidir se o sistema deve irrigar o jardim. Em caso positivo, também é responsável por enviar uma mensagem ao módulo NodeMCU ativando o módulo de irrigação nele implementado.

4. Classe *InterfaceWeb*

   Essa classe implementa a interface gráfica com o usuário. É por meio dela que o usuário visualiza os dados dos processos realizados pelo sistema, e também altera o modo de operação. Ela possui três atributos, sendo eles:

   - **data**: data e hora da última irrigação realizada pelo sistema. Se ainda não houve irrigações, deve mostrar texto condizente explicando a situação.
   - **previsao**: atributo que mostra qual foi a última previsão do tempo recebida pelo sistema. Se está em modo automático, não deve mostrar previsão.
   - **volume**: quantidade de água (em litros) utilizada na última irrigação do sistema. Se ainda não houve irrigação, mostra texto condizente.

   E também uma operação:

   - **altera_modo**: essa operação altera o modo de operação do sistema entre modo inteligente e modo automático, a partir do comando do usuário.

A implementação do sistema é então feita levando a partir dos projetos físico e estrutural do sistema.

4 Implementação

A implementação do sistema proposto foi feita em três frentes: o módulo NodeMCU, o Raspberry Pi e a interface gráfica. Para o desenvolvimento da lógica de controle dos componentes físicos, utilizou-se o ambiente Arduino IDE e linguagem C++, em sistema Windows, para o NodeMCU. A interface gráfica, bem como a implementação do servidor, foram desenvolvidos a partir do *Visual Studio Code*, em sistema *Linux*, utilizando as linguagens de programação HTML e *Python*, respectivamente.

A frente do módulo NodeMCU contempla os sensores e atuadores, a conexão Wi-Fi e MQTT e também a solicitação de hora. A frente do Raspberry Pi contempla a implementação do *broker* e cliente do usuário na conexão MQTT, a solicitação à API de clima e seu processamento, e a lógica de controle utilizada no modo inteligente. Por fim, a frente da interface abrange a visualização dos dados referentes às irrigações e também a troca do modo de operação do sistema.

Para cada frente, são apresentados os métodos utilizados e detalhamento de como foram feitas essas implementações em cada uma delas.

4.1 Módulo NodeMCU

Nesse componente, é feita a leitura de sensores e conversão desses valores lidos a valores interpretáveis, e o controle do atuador. Também foi implementada um método de requisição do horário atual. Nessa frente, por último é necessário implementar as trocas de mensagens com o servidor, via protocolo MQTT. Para isso, primeiramente deve-se implementar a conexão Wi-Fi, e depois define-se os tópicos e mensagens trocadas.

4.1.1 Sensores e atuadores

Os sensores do sistema são o sensor de umidade do solo e sensor de fluxo. Já o atuador é a válvula solenoide.

4.1.1.1 Leitura do sensor de umidade do solo

O sensor de umidade é conectado ao pino analógico A0 do módulo NodeMCU, e fornece valores entre 1024, indicando solo totalmente seco, e 0, indicando solo muito úmido. Desse modo, é feito um mapeamento dos valores para 0 a 100, para que a leitura seja feita em porcentagem.

Para obter uma maior precisão na leitura e atenuar erros causados por ruídos, é feita uma média entre 5 leituras realizadas com intervalos de 10 milissegundos entre elas.

4.1.1.2 Leitura do sensor de fluxo

A leitura deste sensor é feita a partir da contagem de pulsos elétricos gerados pelo sensor de efeito Hall. Para contar esses pulsos, é utilizada uma interrupção. A interrupção funciona da seguinte maneira: depois de configurar um pino de interrupção, cada vez que o módulo detecta o nível HIGH em tal pino, a função principal de *loop* é interrompida, e o controlador executa a função de interrupção. Deste modo, basta programar a função de interrupção como um contador de pulsos para obter o fluxo de água.

Deve-se gravar a função de interrupção na memória *RAM*, visto que ela pode ocorrer durante um acesso à memória *Flash*, e tentar acessar uma interrupção gravada na *Flash* ao mesmo tempo pode fazer o sistema entrar em pane.

O *datasheet*[1] do sensor de fluxo fornece alguns dados de calibração e a curva do fluxo de água Q' (em L/H) em função da frequência de pulsos f (em Hz). Assumindo um comportamento linear no domínio especificado, uma regressão linear fornece a seguinte equação:

$$Q' = 7,779f - 15,283$$

Fazendo a conversão para L/min:

$$Q = 0,1296f - 0,2539$$

Levando-se em consideração o erro indicado de $\pm 5\%$, adota-se para este sensor um fator de correção de $7,5$ min/(Ls), ou seja:

$$Q = \frac{f}{7,5}$$

Sendo c o contador de pulsos do sensor durante o intervalo de tempo Δt (em s) considerado, a frequência f (em Hz) é calculada da seguinte forma:

$$f = \frac{c}{\Delta t}$$

O fluxo Q (em L/min) é então determinado por:

$$Q = \frac{f}{7,5} = \frac{c}{7,5\Delta t}$$

[1] Disponível em: <http://www.mantech.co.za/Datasheets/Products/YF-S201_SEA.pdf>

Convertendo o fluxo q para L/s:

$$q = \frac{Q}{60}$$

Por fim, o volume total V (em L) pode ser calculado por:

$$V = q\Delta t$$

O dimensionamento do volume de água para irrigação deve ser feito de acordo com o tamanho do jardim. Para esta implementação, foi determinado um volume fixo de $0,5$ L por irrigação.

Para validar a calibração, utilizou-se um recipiente com capacidade de meio litro com água, observando o valor indicado. Esse experimento foi repetido 10 vezes. Os resultados estão apresentados na Tabela 4.1.

Tabela 4.1 – Resultados do teste do sensor de fluxo.

Teste Nº	Volume (L)
1	0,49
2	0,47
3	0,51
4	0,42
5	0,58
6	0,42
7	0,52
8	0,45
9	0,53
10	0,49
Média	0,49
Desvio	0,05

Fonte: autoria própria.

Os resultados mostram uma medição muito próxima da quantidade desejada e uma média de 0,49L e um desvio-padrão de 0,05L. Assim, considerou-se a calibração do sensor adequada para o propósito deste trabalho, visto que o sistema não exige uma precisão elevada.

Uma observação que pode ser feita é que, no projeto, existe um atraso entre a passagem de água pela válvula e a sua contabilização pelo sensor de fluxo, tanto pela distância física entre os componentes quanto pelo tempo computacional envolvido. Dessa maneira, é possível que o volume de água utilizado na irrigação seja ligeiramente maior do que o $0,5$L estipulado.

4.1.1.3 Controle da válvula

A válvula solenoide deve ser acionada apenas no momento da irrigação, e deve ser fechada quando a quantidade de água utilizada seja equivalente ao volume determinado de $0,5$L.

O controle do módulo relé é simples, por meio de um pino digital que abre ou fecha o circuito do relé. O evento que controla a abertura da válvula é o recebimento de uma mensagem do servidor indicando que deve haver irrigação. O fechamento da válvula é controlado pelo *feedback* da leitura do sensor de fluxo.

No modo inteligente, a mensagem do servidor é enviada apenas se o processamento dos dados levar o sistema à decisão de que a irrigação é necessária. Já no modo automático, o módulo controlador recebe a mensagem diariamente, independentemente das leituras de sensor e da previsão do tempo. O comportamento de fechamento da válvula se mantém o mesmo em ambos os casos.

4.1.2 Data e hora

A operação do sistema é definida para ocorrer uma vez ao dia. Sendo assim, é necessário pré-determinar um horário para que ela ocorra. Assim, foi escolhido um intervalo diário de 30 minutos, e se faz a verificação de hora uma vez a cada 30 minutos, garantindo que será realizada exatamente uma vez ao dia.

Para contabilizar os milissegundos desde que o programa começou, utilizou-se o método integrado:

```
1 millis()
```

Optou-se por esse método porque ele permite a realização de atividades em paralelo durante seu tempo de espera, como receber mensagens via protocolo MQTT. Ele utiliza uma variável do tipo *unsigned long*, o que é capaz de contabilizar aproximadamente 50 dias. Essa verificação é feita a cada 30 minutos e se verifica se o horário está entre o horário pré-estipulado para ocorrer a irrigação.

A sincronização com o horário brasileiro é feita por meio do protocolo NTP. O cliente de tempo que é instanciado faz uma chamada ao servidor de endereço <*pool.ntp.org*>. De acordo com a documentação do *NTP Pool Project*[2], fazendo uma requisição ao endereço do servidor mencionado, o sistema tentará encontrar o servidor mais próximo da localização física do cliente.

Essa requisição retorna a data e hora oficial UTC+0. Desse modo, assim ainda é necessário converter ao horário do Brasil, que é UTC-3. É necessário então subtrair o

[2] Disponível em: <https://www.ntppool.org/en/>.

equivalente em segundos do tempo de 3 horas da resposta obtida. Isso é feito da seguinte maneira:

```
1 WiFiUDP ntpUDP;
2 const long utcOffset = -3*60*60;
3 NTPClient timeClient(ntpUDP, "pool.ntp.org", utcOffset);
4 timeClient.begin();
5 timeClient.update();
```

O método **begin** inicia o cliente de tempo, e o método **update** sincroniza o cliente com o servidor a cada vez que é executada.

4.1.3 Conexão Wi-Fi

A conexão com a rede Wi-Fi é realizada utilizando a biblioteca *ESP8266WiFi*[3]. Ela possui a função de conectar a placa ESP8266 ao Wi-Fi, e foi desenvolvida com base na biblioteca de mesma funcionalidade para Arduino (GROKHOTKOV, 2017).

A conexão ao Wi-Fi é feita pelo seguinte linha de código:

```
1 WiFi.begin("network-name", "pass-to-network");
```

É necessário alterar os campos **network-name** e **pass-to-network** pelo nome do SSID e senha da rede utilizada. Após essa configuração, o módulo NodeMCU se conecta à rede.

4.1.4 Configuração do protocolo MQTT

Para implementar a troca de mensagens pelo protocolo MQTT, utilizou-se a biblioteca *PubSubClient*[4]. De acordo com o autor dessa biblioteca, Nick O'Leary, ela implementa um cliente para publicar e receber mensagens com um servidor que suporta o protocolo MQTT.

Sendo assim, inicialmente é necessário criar a instância de um cliente. Isso é feito a partir das linhas de código:

```
1 WiFiClient espClient;
2 PubSubClient client(espClient);
```

Na primeira linha, cria-se uma instância capaz de se conectar à rede Wi-Fi, chamada **espClient** e, em seguida, a instância do cliente, chamada **client**. Em seguida, é necessário configurar o cliente para se conectar ao servidor do Raspberry. Isso é feito a partir da seguinte linha:

[3] Documentação disponível em: <https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>.

[4] Documentação disponível em: <https://pubsubclient.knolleary.net/api>.

```
client.setServer(mqtt_server, 1883);
```

Em que **mqtt_server** é o número do IP do Raspberry que atua como *broker*, e 1883 é a porta padrão reservada para o uso do protocolo MQTT.

Depois de configurado, o cliente deve se conectar ao *broker* utilizando uma identificação, ou ID, própria e, caso configurados, o nome do usuário e senha do MQTT:

```
client.connect(ID, user, pass);
```

A função **connect()** retorna um código, que pode ser de sucesso ou fracasso. Caso a conexão tenha falhado, uma boa prática é tentar conectar novamente. Também é importante verificar a conexão de tempos em tempos, para garantir que o cliente esteja conectado.

Então, o cliente pode publicar mensagens e fazer uma assinatura de tópicos. Isso é feito, respectivamente, a partir dos comandos:

```
client.publish(topic, msg);
```

e

```
client.subscribe(topic);
```

No projeto, são implementados 4 tópicos:

1. **esp8266/modo_operacao**: o cliente assina esse tópico que controla qual o modo de operação que o módulo NodeMCU deve atuar (inteligente ou automático).
2. **esp8266/irrigar**: o cliente assina esse tópico para saber se o jardim deve ser irrigado.
3. **esp8266/umidade**: o cliente publica a leitura da umidade do solo.
4. **esp8266/volume**: o cliente publica o volume de água usado.

Por fim, uma *callback* é uma função chamada quando o cliente recebe uma mensagem de algum tópico que assinou. Essa função recebe o tópico, a mensagem e seu comprimento como parâmetros. Sua lógica deve ser programada para que o controlador realize diferentes atividades de acordo com qual foi o tópico e conteúdo da mensagem. A *callback* é configurada da seguinte maneira:

```
void callback(topic, msg, len) {
    ...
}
client.setCallback(callback);
```

Dessa maneira, o cliente configurado está apto a receber e enviar mensagens via protocolo MQTT.

4.2 Raspberry Pi

Nessa seção são apresentados os métodos e soluções utilizadas para implementar um servidor e cliente no Raspberry Pi e também o processamento para a lógica de controle de decisão para irrigação.

4.2.1 Configuração do protocolo MQTT

No Raspberry Pi, há a necessidade de implementar um servidor e um cliente MQTT. O servidor tem a função de encaminhar as mensagens recebidas dos clientes. Como o Raspberry também tem função de enviar, receber e processar mensagens, deve atuar como cliente.

4.2.1.1 Servidor

O *broker* MQTT é implementado no Raspberry Pi usando o *Mosquitto*[5], um *software open-source* e leve, adequado para uso em uma diversidade de dispositivos diferentes. Ele deve ser o primeiro a ser inicializado e se manter conectado para poder fazer a comunicação com os clientes.

O projeto *Mosquitto* também possui dois comandos de utilidades, **mosquitto_pub** e **mosquitto_sub**, ideais para serem usados em ambientes de desenvolvimento e testes. Eles implementam, respectivamente, um cliente que publica mensagens a um tópico e um cliente que faz assinatura de um tópico. Esses comandos foram utilizados para fazer a validação da implementação dos outros clientes, verificando se recebem e publicam mensagens de acordo com o comportamento estabelecido para o sistema.

4.2.1.2 Cliente

Já o cliente no Raspberry é usado tanto para fazer a integração da interface gráfica com as mensagens publicadas e recebidas quanto para enviar mensagens que não incorporam a interface (como a decisão de irrigar o jardim) ao cliente que assina os respectivos tópicos.

Para implementar o cliente, utilizou-se o *Eclipse Paho*[6], um *software open-source* que implementa um cliente MQTT em uma variedade de linguagens de programação. Para esta implementação, utilizou-se a linguagem *Python*.

Inicialmente, assim como o cliente anterior implementado no módulo NodeMCU, é preciso instanciar o cliente. Sua inicialização não requere nenhum parâmetro obrigatório, e, caso não seja atribuído um ID, este campo é gerado aleatoriamente.

[5] Documentação disponível em: <https://mosquitto.org/documentation/>.
[6] Documentação disponível em: <https://www.eclipse.org/paho/index.php?page=clients/python/docs/index.php>

```
mqttc = mqtt.Client()
```

É necessário configurar também as ações (*callbacks*) que devem ser tomadas no momento em que o cliente se conecta e no momento em que recebe uma mensagem. É possível configurar outras *callbacks*, porém no escopo deste projeto não é necessário.

Ao se conectar, o cliente deve assinar os tópicos de interesse. Isso é feito da seguinte maneira:

```
def on_connect(...):
    client.subscribe(topico1)
    client.subscribe(topico2)
    ...
mqttc.on_connect = on_connect
```

Para o caso de o cliente assinar diversos tópicos, o *Paho* permite definir diferentes funções e as atribuir, respectivamente, aos tópicos assinados. Isso é feito por meio do método já implementado na biblioteca **message_callback_add()**

```
def on_message_topico1(...):
    ...
def on_message_topico2(...):
    ...
mqttc.message_callback_add(topico1, on_message_topico1)
mqttc.message_callback_add(topico2, on_message_topico2)
```

Depois das configurações, o cliente deve ser conectado e executado em *loop* para permitir o envio e recebimento de mensagens a qualquer momento.

4.2.2 API *OpenWeather*

A API escolhida para buscar informçações da previsão do tempo foi o *OpenWeather*[7]. Para obter a previsão do tempo, é necessário fazer uma requisição HTTP ao servidor da API.

Para isso, é necessário especificar a localização da qual se deseja obter os dados e também uma *API key*, chave para uso pessoal. A localização é fornecida através da latitude e longitude da cidade. Para a cidade de São Paulo, esses parâmetros estão definidos na Tabela 4.2, de acordo com a documentação.

Desse modo, a chamada é feita da seguinte maneira, substituindo os campos indicados entre { }:

```
api.openweathermap.org/data/2.5/weather?lat={lat}&...
```

[7] Documentação disponível em: <https://openweathermap.org/api/one-call-api>

Tabela 4.2 – Parâmetros da API para a cidade de São Paulo.

id	3448439
name	São Paulo
lat	-23.547501
lon	-46.636108

Fonte: *OpenWeather*.

```
...lon={lon}&appid={API key}
```

O objeto retornado é do tipo JSON. Como não é possível selecionar na requisição os campos desejados, faz-se então um pequeno processamento dos dados obtidos para extrair apenas a previsão do tempo do dia atual e do dia seguinte.

4.2.3 Lógica de controle

Quando o processador obtém o valor da umidade do solo e a previsão do tempo, precisa decidir se o jardim deve ou não ser irrigado. A lógica utilizada para esse processamento é apresentada na Figura 4.1. No caso de irrigar, publica uma mensagem pelo respectivo tópico indicando tal decisão.

Inicialmente, o programa verifica se a previsão para o dia atual é de chuva. Em caso positivo, não precisa irrigar. Em caso negativo, verifica tanto se a previsão para o dia seguinte é de chuva quanto se a umidade do solo está em 50% ou mais. Novamente, em caso positivo, não há necessidade de irrigar. Por último, em caso negativo, verifica se a umidade do solo está maior que 70%. Se nenhuma das condições é cumprida, o jardim é irrigado.

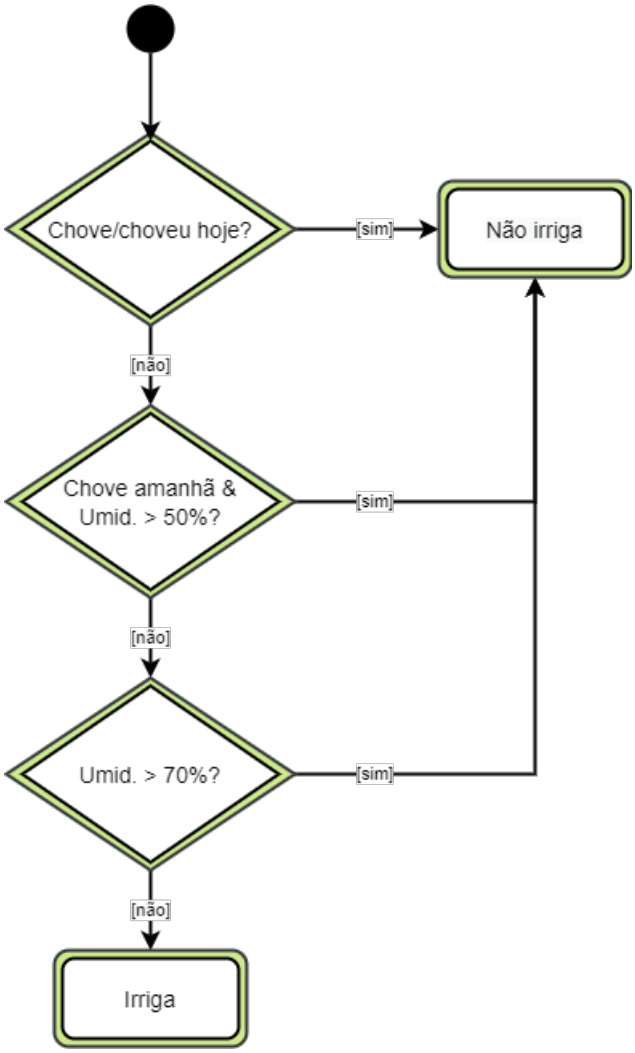
Para validar a implementação, foi desenvolvido e aplicado também um *script* de teste, que testa todos os 12 casos, em que as variáveis são:

1. **chove_hoje**: 0 (não) ou 1 (sim).
2. **chove_amanha**: 0 (não) ou 1 (sim).
3. **umidade**: $u \leq 50$, $50 < u \leq 70$ ou $u > 70$. No teste, utilizou-se os valores arbitrários $u = 40$, $u = 60$ e $u = 80$.

Como esperado, os três casos em que a irrigação deve ser acionada são:

$$\{\text{chove\_hoje}, \text{chove\_amanha}, u\} = [\{0, 1, 40\}, \{0, 0, 60\}, \{0, 0, 40\}]$$

Figura 4.1 – Lógica de controle utilizada na decisão.



Fonte: autoria própria.

4.3 Interface gráfica

A interface com o usuário foi implementada por meio de um conjunto de páginas *Web*, em linguagem HTML, ligadas por *links*. Ao todo, foram implementadas três páginas: a página inicial (*home.html*), a página de consulta de dados (*consulta.html*) e a página de modos de operação (*modos.html*).

A página inicial contém um *Widget* com a previsão do tempo, e botões que redirecionam às outras páginas. O *OpenWeather* possui um construtor de *widgets*, no qual é possível obter o código para uma interface já pronta com a previsão do tempo, bastando inserir a localidade e a chave *API key*. A Figura 4.2 mostra o componente selecionado para compôr a página.

Além disso, todas as páginas possuem um menu lateral com o mapa da aplicação, sendo possível clicar sobre elas para ser redirecionado.

Figura 4.2 – *Widget* de previsão obtido por meio da API.

Fonte: *OpenWeather*.

A página de consulta de dados fornece as três informações básicas: volume utilizado na última irrigação, data e hora em que ela ocorreu. Caso não tenha ocorrido nenhuma, a página não mostra nenhuma informação.

Por último, a página de modos de operação contém uma breve informação sobre os dois modos de operação do sistema (inteligente e automático), e um botão que alterna entre os dois, indicando qual é o modo ativo no momento.

Para renderizar as páginas, foi utilizado o *Flask*[8], um *framework web*, ou seja, um *software* usado para suportar o desenvolvimento *web*. Ele é escrito em *Python* e voltado para o desenvolvimento de projetos simples, como uma página *web* básica.

Para utilizá-lo, é necessário definir as rotas da aplicação e as respectivas funções que serão executadas ao acessar essas rotas. Isso é feito por meio do decorador **@route()**. Para renderizar uma página, utiliza-se a função integrada **render_template()**.

[8] Documentação disponível em: <https://flask.palletsprojects.com/en/1.1.x/api/>.

5 Resultados e discussões

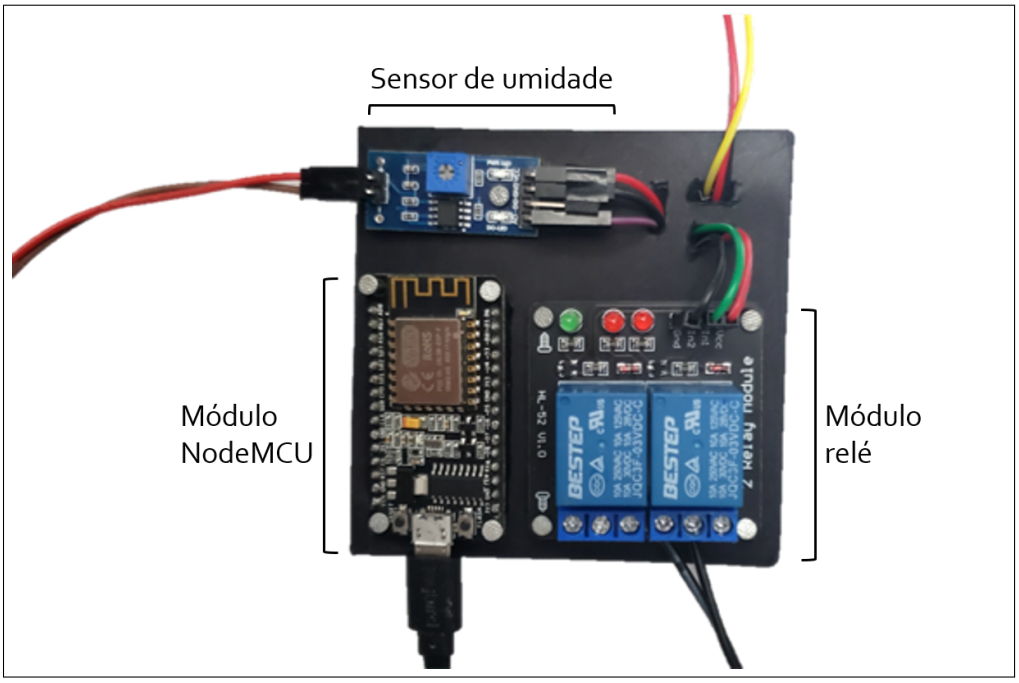
Nesse capítulo são apresentados os resultados da implementação do sistema e dos testes realizados, bem como discussões pertinentes a todo o processo de desenvolvimento do projeto.

5.1 Resultados

A montagem do sistema é apresentada a seguir. A Figura 5.1 apresenta o resultado da montagem do módulo NodeMCU, com destaque para o módulo relé e parte do sensor de umidade.

A montagem completa do sistema é apresentada na Figura 5.2, que contém a os componentes eletrônicos e hidráulicos, e na Figura 5.3 mostra o sistema em operação, juntamente com uma planta.

Figura 5.1 – Montagem dos componentes eletrônicos.

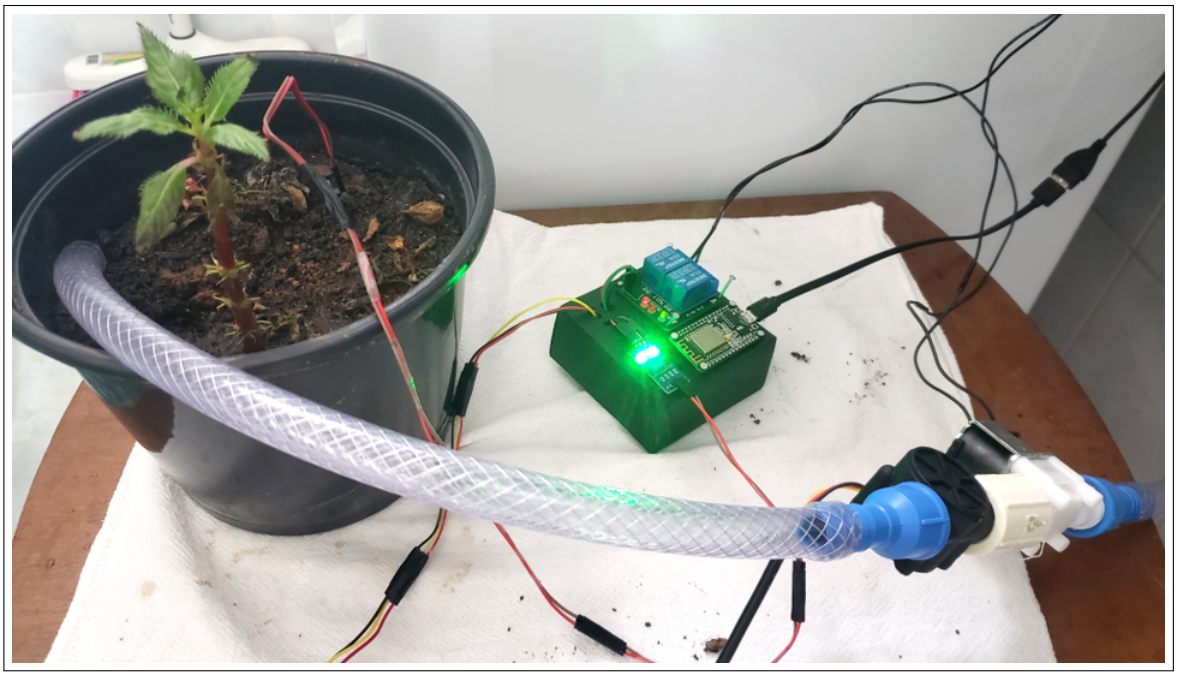


Fonte: autoria própria.

Além disso, houve a implementação da interface gráfica para interação com o usuário. Ela é feita em três páginas diferentes. As Figuras 5.4, 5.5 e 5.6 mostram esses resultados na versão *desktop*.

Já as Figuras 5.7, 5.8a e 5.8b apresentam o resultado da mesma implementação, porém visualizadas na versão *mobile*.

Figura 5.2 – Montagem eletrônica e hidráulica.



Fonte: autoria própria.

Figura 5.3 – Sistema integrado com uma planta.

Fonte: autoria própria.

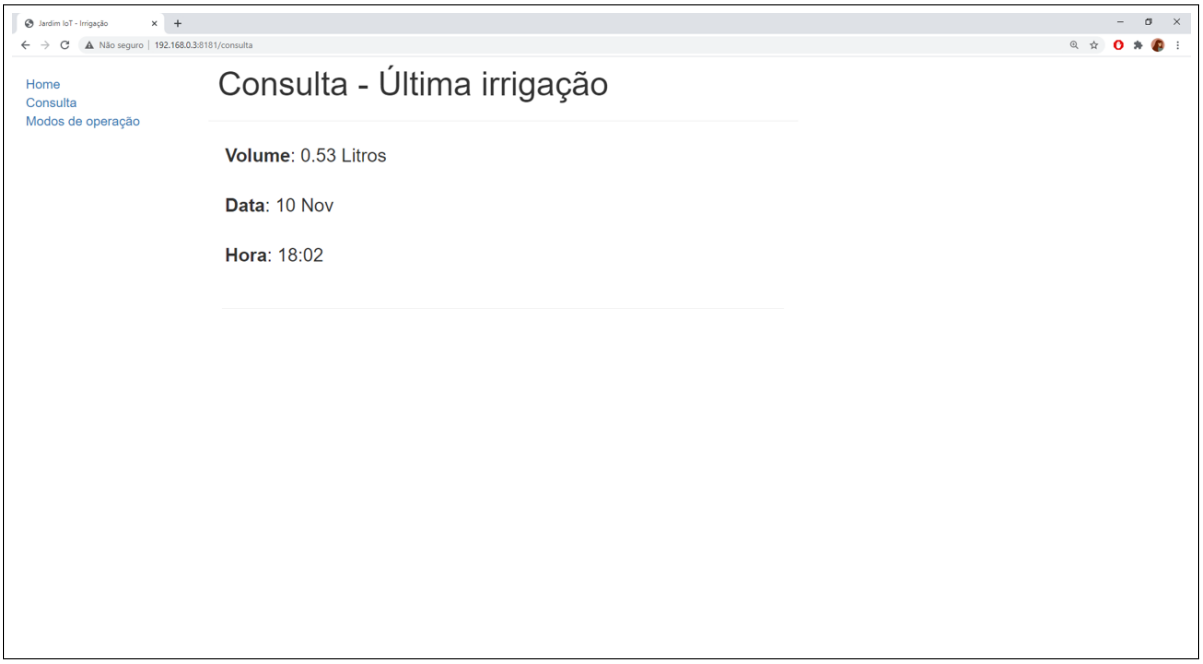
Como a implementação é feita localmente, apenas os dispositivos com acesso à rede Wi-Fi possuem acesso às páginas desenvolvidas. Se as páginas fossem hospedadas em um servidor externo, qualquer dispositivo com acesso à internet teria acesso a elas.

Após a montagem e execução do projeto, foi possível verificar que o sistema se comportou de acordo com o modelo especificado.

Figura 5.4 – Página inicial na versão *desktop*.

Fonte: autoria própria.

Figura 5.5 – Página de consulta de dados na versão *desktop*.

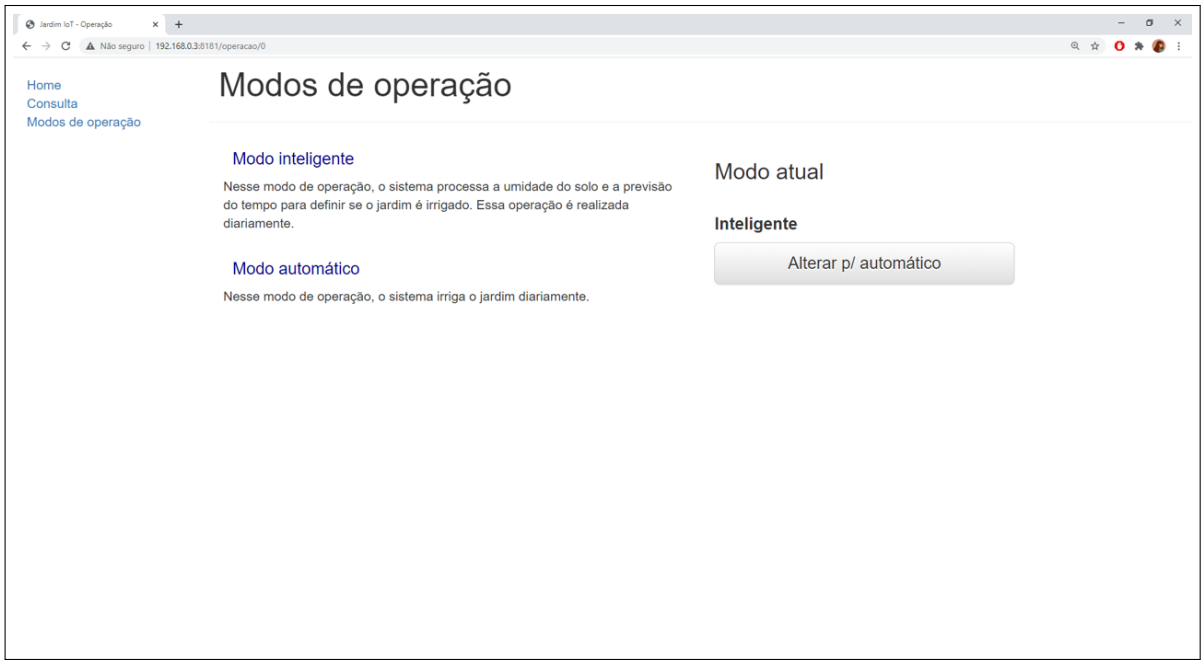


Fonte: autoria própria.

5.2 Discussões

Em relação à implementação do sistema, outra maneira possível de limitar a quantidade de água na irrigação seria através do medidor de umidade: irrigar o solo até que a umidade atinja certo valor. Porém, isso pode ser susceptível a mais erros pois o solo demora a absorver a água proveniente da irrigação, levando a um mal dimensionamento

Figura 5.6 – Página de modos de operação na versão *desktop*.



Fonte: autoria própria.

do sistema. Também é possível limitar por tempo, ou seja, a irrigação deve durar um certo período de tempo e depois cessar.

No que diz respeito à interface gráfica, para uma melhor experiência do usuário, é possível desenvolver aplicações *mobile* para construir a interface visual. Já existem algumas soluções prontas disponíveis, como exemplos pode-se citar Node-RED, Blynk e ThingsBoard. Essas plataformas, além de serem ferramentas de desenvolvimento para a comunicação entre os dispositivos, possuem integradas interfaces gráficas para o usuário. Neste trabalho, optou-se por não utilizar essas ferramentas.

Por fim, o fato de o módulo NodeMCU precisar estar disponível para receber mensagens a qualquer momento o impede de entrar no modo *Deep Sleep*, em que o rádio Wi-Fi, CPU e CLOCK são desligados. Sendo assim, existe um *trade-off* entre a possibilidade de alterar o modo de operação do sistema e a economia de energia gerada por colocar o módulo em espera.

Figura 5.7 – Página inicial na versão *mobile*.

Fonte: autoria própria.

Figura 5.8 – Páginas na versão *mobile*.

(a) Consulta de dados.

(b) Modos de operação.

Fonte: autoria própria.

6 Conclusão

Um dos potenciais da Internet das Coisas é a capacidade de conectar diversos dispositivos e sensores que transmitem dados em tempo real por meio da Internet. Essas informações em tempo real oferecem vantagens, pois tornam possíveis análises instantâneas dos dados e aumentam a velocidade e eficiência nas tomadas de decisões que levam em conta esses dados.

Levando isso em consideração, e também que atualmente se desperdiça uma quantidade relevante de água no país, a intenção desse trabalho foi desenvolver um sistema autônomo completo de irrigação de jardim com acesso à internet. Assim, os dados obtidos por sensores puderam ser transmitidos e recebidos por outras máquinas, da mesma maneira que o sistema é capaz de receber dados externos. A capacidade dessa transmissão de dados é utilizada para gerar economia de água.

O trabalho contemplou todas as fases do projeto, envolvendo pesquisa, revisão da literatura, modelagem do comportamento do sistema, implementação e execução. Ao final, obteve-se o sistema desejado.

O propósito do projeto não foi desenvolver um produto final para o mercado, mas sim obter uma prova de conceito de uma tecnologia em expansão a uma aplicação em projeto residencial. A aplicação de tecnologias como Internet das Coisas é muito grande. Este trabalho tratou dessa aplicação na área de automação residencial, porém a modernização e conexão em tempo real de objetos e sensores já é uma realidade existente em diversas outras áreas, como saúde, indústria, urbanização ou varejo.

6.1 Próximos passos

Existem diversas propostas de abordagens para trabalhos futuros que contemplam o tema de irrigação incorporando Internet das Coisas.

Uma proposta de complementação é adaptar a interface e o comportamento do sistema para uma abordagem mais voltada à experiência do usuário, com parâmetros customizáveis, como controle por volume ou tempo (e suas respectivas quantidades), horário e frequência de irrigação.

Outra proposta é implementar um banco de dados e uma inteligência artificial com foco em aprendizado de máquina, que pode ser treinada para prever as épocas mais chuvosas e também a necessidade de água do solo, com o intuito de aumentar a economia de água.

Levando em conta o viés de automação residencial do projeto, pode-se estender a amplitude da automação, integrando mais componentes residenciais voltados ao cuidado da casa por meio da Internet das Coisas, como iluminação, ventilação e umidade do ambiente, e fazendo do Raspberry um servidor central para todos os componentes autômatos.

Por fim, considerando a questão econômica do Brasil, uma última proposta de trabalho futuro é a adaptação do presente projeto para atuar em plantações de larga escala, gerando economia de água em maiores proporções.

Referências

Agência Nacional de Águas (ANA). *Conjuntura dos recursos hídricos no Brasil 2019: Informe anual*. Brasília, 2019. Disponível em: <http://www.snirh.gov.br/portal/snirh/centrais-de-conteudos/conjuntura-dos-recursos-hidricos/conjuntura_informe_anual_2019-versao_web-0212-1.pdf>. Citado na página 20.

AGGARWAL, C. C.; ASHISH, N.; SHETH, A. The internet of things: A survey from the data-centric perspective. In: AGGARWAL, C. (Ed.). *Managing and Mining Sensor Data*. [S.l.]: Springer, 2013. cap. 12. Citado na página 23.

ASHTON, K. That 'internet of things' thing. *RFID Journal*, p. 1, 2009. Citado na página 22.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer Networks*, v. 54, p. 2787–2805, 2010. Citado na página 22.

BEOCK, L.; CONSONE, C.; LIMA, L.; PETRICA, E. *Protocolo HTTP*. [S.l.], Universidade do Estado de Mato Grosso, 2011. Disponível em: <https://img.vivaolinux.com.br/imagens/artigos/comunidade/Protocolo%20HTTP.pdf>. Citado na página 30.

CORREIA, G. R.; ROCHA, H. R. de O.; RISSINO, S. das D. Automação de sistema de irrigação com monitoramento via aplicativo web. *REVENG*, Viçosa, MG, v. 24, n. 4, p. 314–325, 2016. Citado na página 17.

GENGO, R. de C.; HENKES, J. A. A utilização do paisagismo como ferramenta na preservação e melhoria ambiental em área urbana. *Revista Gestão e Sustentabilidade Ambiental*, Florianópolis, v. 1, n. 2, p. 55–81, 2013. Disponível em: <http://www.portaldeperiodicos.unisul.br/index.php/gestao_ambiental/article/view/1206>. Acesso em: 08 fev. 2020. Citado na página 15.

GHIZZI, R. B. *Sistema automatizado de irrigação residencial com reciclagem de águas pluviais*. Dissertação (Trabalho de Conclusão de Curso (Bacharelado em Engenharia Mecatrônica) — Escola de Engenharia de São Carlos - USP, 2016. Citado 2 vezes nas páginas 16 e 17.

GONÇALVES, D.; ARLINDO, J.; PEREIRA, R.; MOURA, V.; JUCA, S. Sistema iot para monitoramento e controle de irrigação. In: *Anais da IV Escola Regional de Informática do Piauí*. Porto Alegre, RS, Brasil: SBC, 2018. p. 310–315. Disponível em: <https://sol.sbc.org.br/index.php/eripi/article/view/5186>. Citado na página 18.

GREHS, D. H. *Sistema de irrigação doméstico baseado em Internet das Coisas*. Dissertação (Trabalho de conclusão de curso (Bacharel em Engenharia de Computação) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2016. Citado 2 vezes nas páginas 17 e 18.

GROKHOTKOV, I. *ESP8266WiFi library*. [S.l.], 2017. Software Documentation. Disponível em: <https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>. Citado na página 56.

GUTIERREZ, R. M. V.; PAN, S. S. K. Complexo eletrônico: automação do controle industrial. *Banco Nacional de Desenvolvimento Econômico e Social*, Rio de Janeiro, n. 28, p. 189–231, 2008. Disponível em: <http://web.bndes.gov.br/bib/jspui/handle/1408/9536>. Acesso em: 17 jun. 2020. Citado na página 20.

JÚNIOR, P. R. T. *Caracterizaćão da rede de sincronizaćão na internet*. Dissertação (Mestrado) — Universidade Federal do Paraná, 2007. Citado na página 31.

KHUSNUTDINOV, A.; USACHEV, D.; MAZZARA, M.; KHAN, A.; PANCHENKO, I. Open source platform digital personal assistant. *32nd International Conference on Advanced Information Networking and Applications Workshops*, 2018. Citado na página 28.

LUCON, C. M. M.; CHAVES, A. L. R. Palestra Horta Orgânica. *O Biológico*, São Paulo, SP, v. 66, n. 1/2, p. 59–62, 2004. Citado na página 15.

MARINO, D. R. D. M.; VASCONCELOS, D. R.; MORAES, S. G. Jardim inteligente IoT - JIIOT. *Rev. Tecnol. Fortaleza*, v. 38, n. 1, p. 39–54, 2017. Citado 2 vezes nas páginas 18 e 19.

MARTINS, I. R.; ZEM, J. L. Estudo dos protocolos de comunicação MQTT e COAP para aplicações machine-to-machine e internet das coisas. *Revista Tecnológica da Fatec Americana*, Americana, v. 3, n. 1, p. 64–87, 2015. Acesso em: 09 fev. 2020. Citado na página 25.

MDN Web Docs. *Uma visão geral do HTTP*. [S.l.], 2017. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>. Citado na página 30.

MILLS, D. L. *Network Time Protocol (NTP)*. [S.l.], 1985. Citado na página 30.

NETO, J. G. Sistemas de irrigação para jardins e gramados. Belo Horizonte, 2017. Disponível em: <https://www.rainbird.com.br/pdf/?urlredirect=/upload/ferramentas-detrabalho/Artigos/Irrigacao-para-Paisagismo.pdf>. Acesso em: 7 fev. 2020. Citado na página 15.

OLIVEIRA, S. de. *Internet das Coisas com ESP8266, Arduino e Raspberry Pi*. 1. ed. São Paulo, SP: Novatec Editora Ltda, 2017. Citado 2 vezes nas páginas 24 e 25.

Oxford University Press (OUP). *Internet of Things*. [s.n.], 2019. Disponível em: <https://www.lexico.com/definition/internet_of_things>. Acesso em: 18 mar. 2020. Citado na página 22.

PARZIALE, L.; BRITT, D. T.; DAVIS, C.; FORRESTER, J.; LIU, W. *TCP/IP Tutorial and Technical Overview*. 8. ed. [S.l.]: IBM, 2006. Citado 2 vezes nas páginas 28 e 29.

PFISTER, C. *Getting Started with the Internet of Things: Connecting Sensors and Microcontrollers to the Cloud*. [S.l.]: O'Reilly, 2011. Citado na página 23.

RODRIGUES, R. A. S.; SOUSA, P. F. C. *Irrigação e drenagem*. Londrina: Editora e Distribuidora Educacional S.A., 2018. 232 p. Citado na página 15.

SANTOS, F. de Assis Martins dos. Projeto de irrigação inteligente. *Holos*, RN, v. 5, n. 26, p. 37–44, 2010. Citado na página 15.

SEIDL, M.; SCHOLZ, M.; HUEMER, C.; KAPPEL, G. *UML @ Classroom*: An introduction to object-oriented modeling. Austria: Sprint International Publishing, 2015. Citado 3 vezes nas páginas 41, 43 e 47.

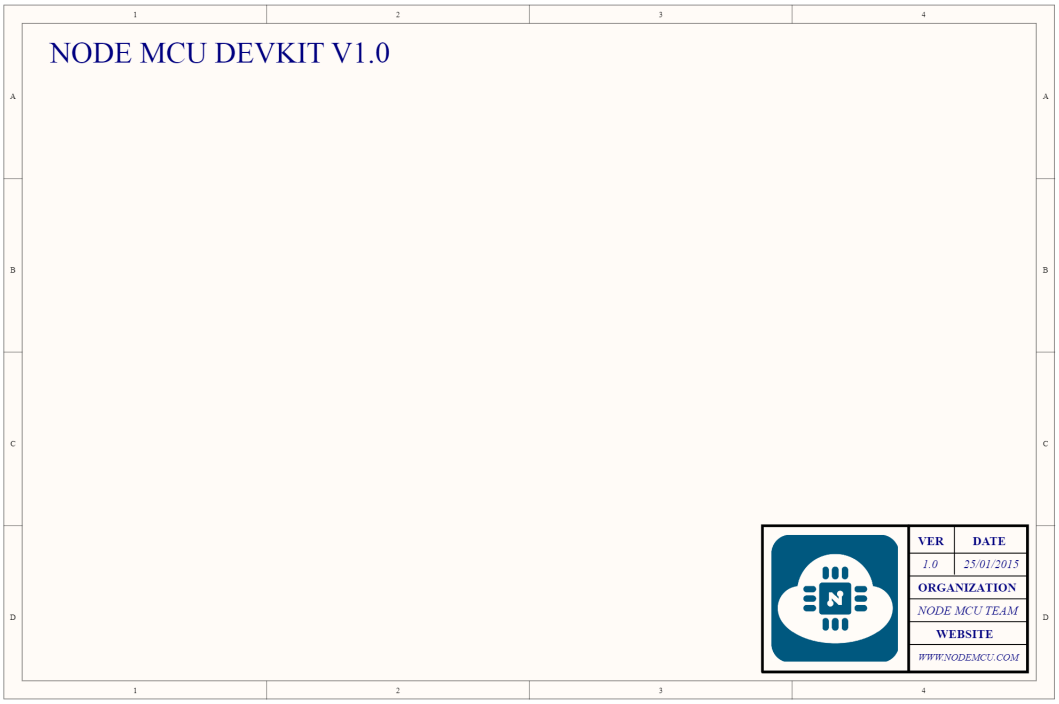
SOMMERVILLE, I. *Engenharia de Software*. São Paulo, SP: [s.n.], 2013. Citado na página 24.

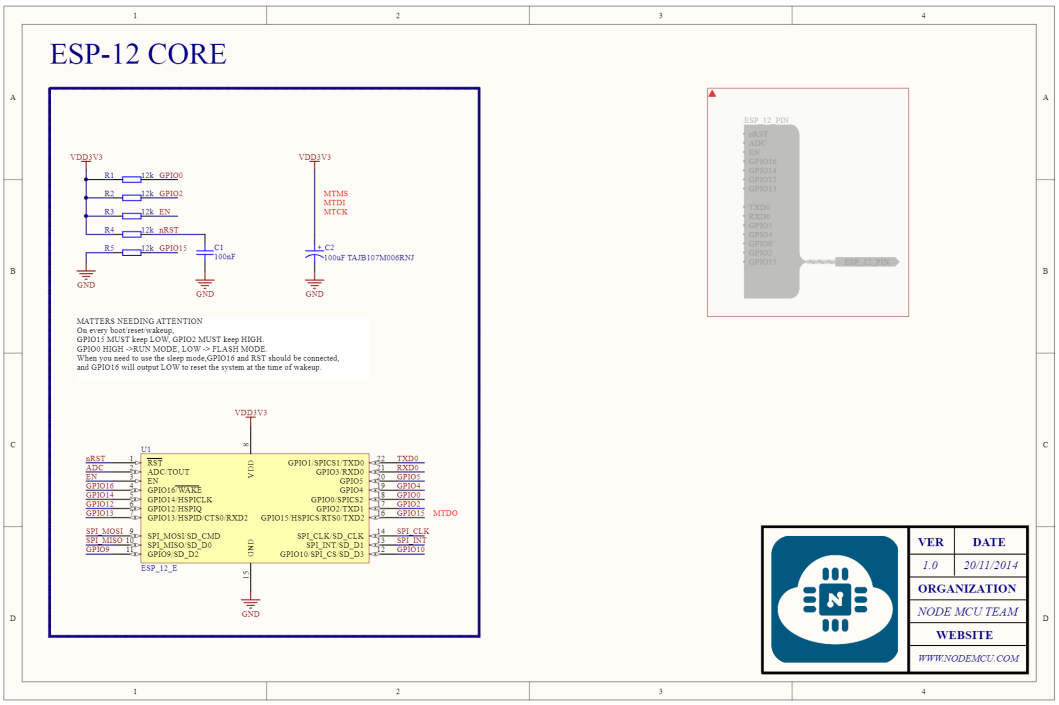
TESTEZLAF, R. *Irrigação: Métodos, Sistemas e Aplicações*. 1. ed. Campinas, SP: [s.n.], 2017. 215 p. Citado na página 15.

YUAN, M. Getting to know mqtt. *IBM Developer*, 2017. Disponível em: <https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/>. Citado 2 vezes nas páginas 25 e 27.

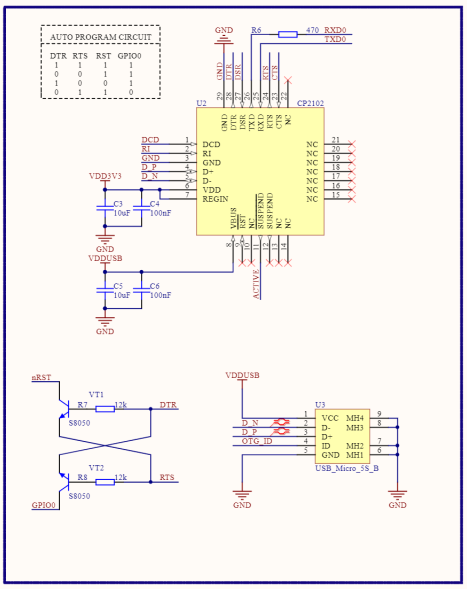
Apêndices

APÊNDICE A – Esquemático do módulo NodeMCU



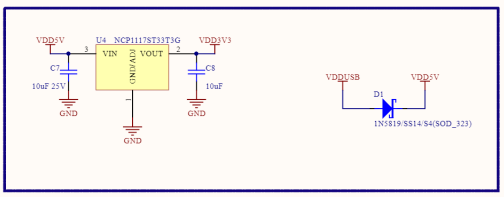


USB TO UART



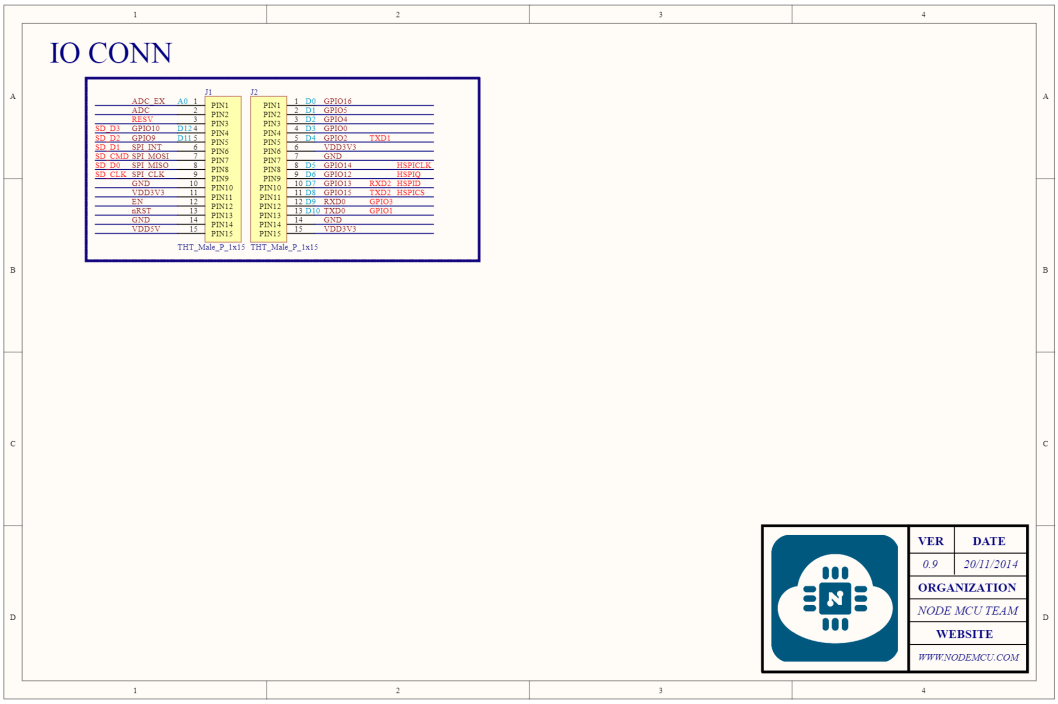
VER	DATE
0.9	20/11/2014
ORGANIZATION	
NODE MCU TEAM	
WEBSITE	
WWW.NODEMCU.COM	

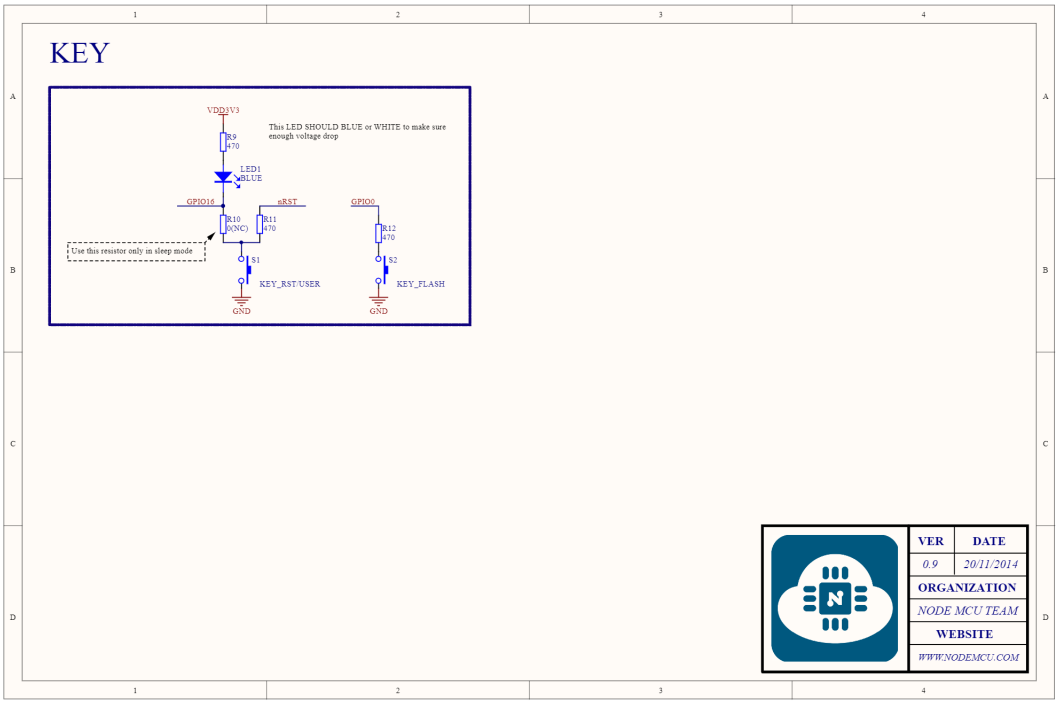
POWER

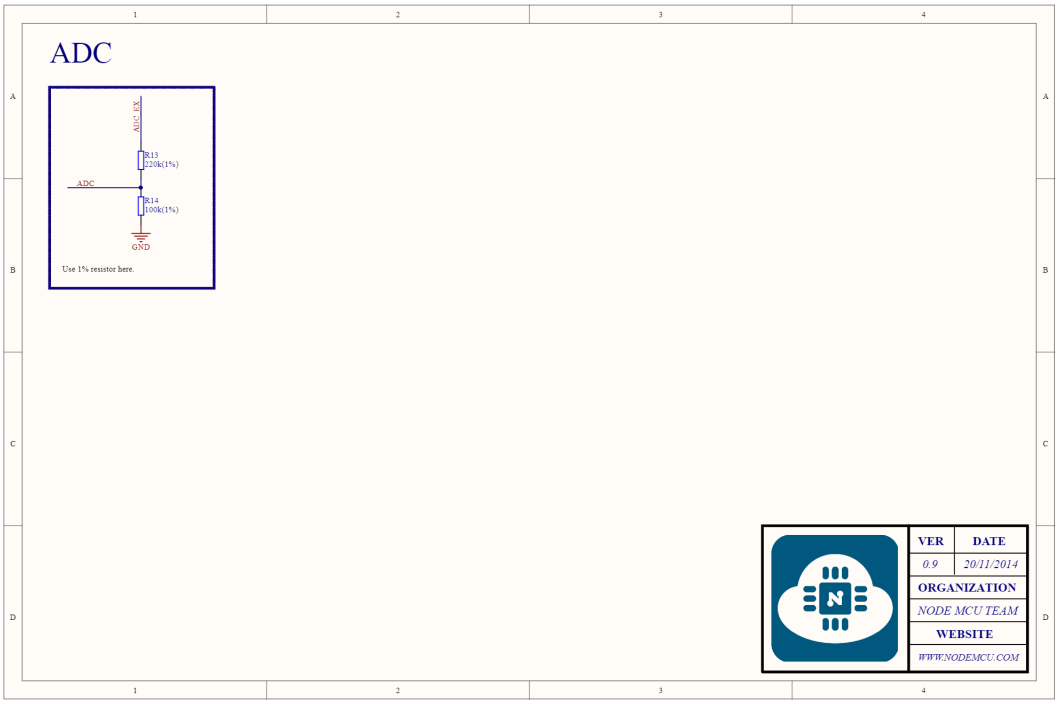


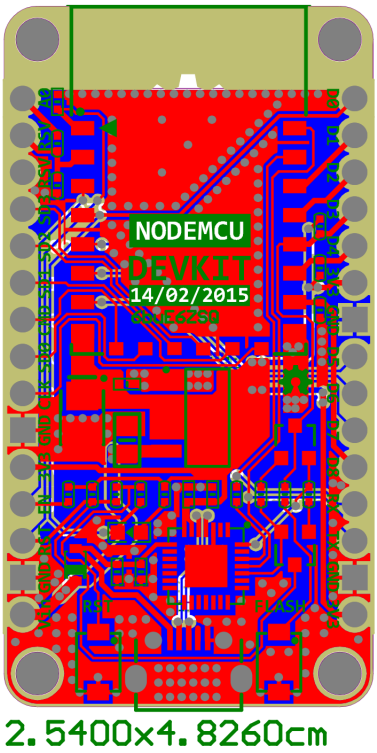
Working Output: 3.3V 800mA
Working Current Limit: 1000mA
Max Current: 1000mA
Max Supply Voltage: 20V
Voltage Dropout:1.2V@800mA

VER	DATE
1.0	28/01/2015
ORGANIZATION	
NODE MCU TEAM	
WEBSITE	
WWW.NODEMCU.COM	


IO CONN
J1
ADC_EX A0 1 PIN1
ADC 2 PIN2
RESV 3 PIN3
SD_D3 GPIO10 D12 4 PIN4
SD_D2 GPIO9 D11 5 PIN5
SD_D1 SPI_INT 6 PIN6
SD_CMD SPI_MOSI 7 PIN7
SD_D0 SPI_MISO 8 PIN8
SD_CLK SPI_CLK 9 PIN9
GND 10 PIN10
VDD3V3 11 PIN11
EN 12 PIN12
nRST 13 PIN13
GND 14 PIN14
VDD5V 15 PIN15
THT_Male_P_1x15
J2
PIN1 1 D0 GPIO16
PIN2 2 D1 GPIO5
PIN3 3 D2 GPIO4
PIN4 4 D3 GPIO0
PIN5 5 D4 GPIO2 TXD1
PIN6 6 VDD3V3
PIN7 7 GND
PIN8 8 D5 GPIO14 HSPICLK
PIN9 9 D6 GPIO12 HSPIQ
PIN10 10 D7 GPIO13 RXD2 HSPID
PIN11 11 D8 GPIO15 TXD2 HSPICS
PIN12 12 D9 RXD0 GPIO3
PIN13 13 D10 TXD0 GPIO1
PIN14 14 GND
PIN15 15 VDD3V3
THT_Male_P_1x15
VER
DATE
0.9
20/11/2014
ORGANIZATION
NODE MCU TEAM
WEBSITE
WWW.NODEMCU.COM


KEY
VDD3V3
R9
470
This LED SHOULD BLUE or WHITE to make sure enough voltage drop
LED1
BLUE
GPIO16
nRST
GPIO0
R10
(XNC)
R11
470
R12
470
Use this resistor only in sleep mode
S1
S2
KEY_RST/USER
KEY_FLASH
GND
GND
VER
DATE
0.9
20/11/2014
ORGANIZATION
NODE MCU TEAM
WEBSITE
WWW.NODEMCU.COM


ADC
ADC_EX
R13
220k(1%)
ADC
R14
100k(1%)
GND
Use 1% resistor here.
VER
DATE
0.9
20/11/2014
ORGANIZATION
NODE MCU TEAM
WEBSITE
WWW.NODEMCU.COM


NODEMCU
DEVKIT
14/02/2015
2.5400x4.8260cm

Bill of Materials

NODE MCU DEVKIT V1.0

Source Data From: NODEMCU_DEVKIT_V1.0.PrjPCB
Project: NODEMCU_DEVKIT_V1.0.PrjPCB
Variant: None

Creation Date: 2015/5/14 12:28:09
Print Date: 14-May-15 12:28:30 PM

Footprint	Comment	LibRef	Designator	Description	Quantity
SMT_C_0402	100nF	SMT_C_0402	C1, C4, C6	Surface mount capacitor 0402	3
SMT_C_Tantalum_B	100uF TAJB107M006RNJ	SMT_C_Tantalum_B	C2	Capacitor, SM Tantalum; Body 3.5 x 2.8 mm (LxW typ)	1
SMT_C_0402	10uF	SMT_C_0402	C3, C5, C8	Surface mount capacitor 0402	3
SMT_C_0805	10uF 25V	SMT_C_0805	C7	Surface mount capacitor 0805	1
SMT_DIODE_SOD_323	1N5819/SS14/S4(SOD_323)	SMT_DIODE_SCHOTTKY_SOD_323	D1	Surface mount schottky diode SOD-323(0805) Package	1
THT_Male_P_1x15-2.54mm	THT_Male_P_1x15	THT_Male_P_1x15	J1, J2	THT Male pin header strip 1x15	2
SMT_LED_0603	BLUE	SMT_LED_0603	LED1	SMT LED	1
SMT_R_0402	12k	SMT_R_0402	R1, R2, R3, R4, R5, R7, R8	Surface mount resistor 0402	7
SMT_R_0402	470	SMT_R_0402	R6, R9, R11, R12	Surface mount resistor 0402	4
SMT_R_0402	0(NC)	SMT_R_0402	R10	Surface mount resistor 0402	1
SMT_R_0402	220k(1%)	SMT_R_0402	R13	Surface mount resistor 0402	1
SMT_R_0402	100k(1%)	SMT_R_0402	R14	Surface mount resistor 0402	1
SMT_SW_PTS_820	KEY_RST/USER	SMT_SW_PTS_820	S1	SMT Tactile ?Switch PTS 820 Series	1
SMT_SW_PTS_820	KEY_FLASH	SMT_SW_PTS_820	S2	SMT Tactile ?Switch PTS 820 Series	1
ESP_12_E_L	ESP_12_E	ESP_12_E	U1	ESP-12 Wi-Fi Module by AI-Thinker	1
CP2102	CP2102	CP2102	U2	CP2102	1
USB_MICRO_5S_B	USB_Micro_5S_B	USB_Micro_5S_B	U3	USB micro female SMT with 4 fixed foot	1
SOT-223	NCP1117ST33T3G	NCP1117	U4	NCP1117, NCV1117, 1.0 A Low-Dropout Positive Fixed and Adjustable Voltage Regulators	1
SMT_TRIODE_NPN	S8050	SMT_TRIODE_NPN	VT1, VT2	Surface mount NPN transistor, package SOT-23	2
					34

Approved **Notes**