Faculdade de Tecnologia de São Paulo

DEPARTAMENTO DE SISTEMAS ELETRÔNICOS

SISTEMA SUPERVISÓRIO DE IRRIGAÇÃO AUTOMATIZADO E DE ACESSO REMOTO

ARTUR ARAUJO PORTO

WELLINGTON BATISTA FERNANDES

São Paulo - SP

2022

Faculdade de Tecnologia de São Paulo

DEPARTAMENTO DE SISTEMAS ELETRÔNICOS

ARTUR ARAUJO PORTO

WELLINGTON BATISTA FERNANDES

SISTEMA SUPERVISÓRIO DE IRRIGAÇÃO AUTOMATIZADO E DE ACESSO REMOTO

Trabalho de Conclusão de Curso apresentado como requisito parcial para a conclusão do curso Superior de Tecnologia em Eletrônica Industrial da Faculdade de Tecnologia de São Paulo.

Orientador: Prof. Dr. Danilo Z. Figueiredo

São Paulo - SP

2022

Ficha Catalográfica

Porto, Artur e Fernandes, Wellington
SISTEMA SUPERVISÓRIO DE IRRIGAÇÃO AUTOMATIZADO E DE ACESSO REMOTO / Porto, Artur e Fernandes, Wellington - 2022.
64 f.
Monografia - Curso Superior de Tecnologia em Eletrônica Industrial - Faculdade de Tecnologia de São Paulo, 2022.
Orientador(a): Prof. Danilo Zucolli Figueiredo
1. Irrigação 2. Sistema Supervisório 3. Protocolo MQTT.

TERMO DE APROVAÇÃO

ARTUR ARAUJO PORTO

WELLINGTON BATISTA FERNANDES

SISTEMA SUPERVISÓRIO DE IRRIGAÇÃO AUTOMATIZADO E DE ACESSO REMOTO

Aprovado em: ___ / ___ / ___

_____

Prof. Dr. Victor Sonnenberg

Coordenador do curso / Chefe do Departamento de Sistemas Eletrônicos

_____

Prof. Dr. Danilo Zucolli Figueiredo

Orientador

_____

Prof. Dr. Roberto K. Yamamoto

Avaliador

_____

Prof. Dr. Victor Sonnenberg

Avaliador

Aluno(a): **Nº Matrícula:**

Título do Trabalho:

Data de Apresentação: **Horário:**

Local:

Orientador(a):

Nota

Membros da Banca: ***Prof.*** _____

Prof. _____

Prof. _____

Nota mínima para aprovação ≥ 6,0

Convidados: _____ _____

Observação da Banca:

_____

Ciente:

_____

Prof. Me.

_____

Prof. Me.

_____

Prof. Dr.

Dedico este trabalho, primeiramente a Deus, por ser essencial em minha vida, Autor de meu destino, meu guia e socorro presente na hora da angústia.

À minha noiva Brunna Nepomuceno Colatto, que com muito carinho e apoio, não mediu esforços para qve eʊ chegasse até esta etapa da minha vida.

Wellington Batista Fernandes

AGRADECIMENTOS

A Deus, por permitir que nós tivéssemos saúde e determinação.

Às nossas famílias que nos incentivaram nos momentos difíceis e compreenderam a nossa ausência enquanto nos dedicávamos à realização deste trabalho.

Aos nossos professores que nos guiaram ao aprendizado; à FATEC pelo desenvolvimento acadêmico; e ao orientador, Prof. Dr. Danilo Zucolli Figueiredo, pelas correções e ensinamentos que nos permitiram apresentar um melhor desempenho no processo de formação acadêmica.

Educação não transforma o mundo.
Educação muda as pessoas.
Pessoas transforam o mundo.

Paulo Freire (1921 - 1997)

LISTA DE FIGURAS

LISTA DE SIGLAS

Sigla - Descrição - Tradução (em caso de língua estrangeira)

Embrapa - Empresa Brasileira de Pesquisa Agropecuária
GPIO - *General Purpose Input/Output* – Entrada/Saída de Uso Geral
HTTP - *HyperText Transfer Protocol* - Protocolo de Transferência de Hipertexto
IBGE - Instituto Brasileiro de Geografia e Estatística
IBM - *International Business Machines Corporation* - Corporação Internacional de Máquinas de Negócios
IDE - *Integrated Development Environment* - Ambiente de Desenvolvimento Integrado
INCRA - Instituto Nacional de Colonização e Reforma Agrária
IOT - *Internet of Things* - Internet das Coisas
M2M - *Machine to Machine* – Máquina para Máquina
MQTT - *Message Queuing Telemetry Transport* - Transporte de Filas de Mensagem de Telemetria
OASIS - *Organization for the Advancement of Structured Information Standards* - Organização para o Avanço de Padrões em Informação Estruturada
PCB - *Printed Circuit Board* - Placa de Circuito Impresso
TCP - *Transmission Control Protocol* - Protocolo de Controle de Transmissão
UFRGS - Universidade Federal do Rio Grande do Sul
USB - *Universal Serial Bus* - Porta Serial Universal

RESUMO

Este trabalho teve por objetivo mostrar as implicações quantitativas da aplicação de protocolos de comunicação a um sistema automatizado de gestão de água e irrigação, como objeto deste estudo, o protocolo MQTT. A abordagem qualitativa foi baseada no desenvolvimento de um protótipo, que torna viável a validação de aplicabilidade do projeto. A pesquisa bibliográfica foi realizada por meio de um estudo exploratório sobre os conceitos de automação, conectividade, democratização de tecnologias e uso consciente de água, possibilitando, assim, a discussão teórica deste trabalho. A partir da análise da literatura, foi possível observar a importância da agricultura familiar na economia brasileira. Uma de suas características mais interessantes é a pouca mecanização dos processos de plantio e colheita, sendo uma ótima oportunidade para o desenvolvimento da modernização dos processos envolvidos, como é o caso da irrigação, motivo deste projeto. Diante dos resultados obtidos, pode-se observar o potencial de integração que o protocolo de comunicação proporciona ao sistema de irrigação, mostrando eficiência na troca de informações coletadas em campo e ainda oferecendo suporte a aplicações que poderão ser implementadas futuramente como o controle de parâmetros inerentes ao código fonte pela troca de informações via MQTT.

Palavras-chave: Irrigação, Sistema Supervisório, Protocolo MQTT.

ABSTRACT

This work aims to show the quantitative implications of the application of communication protocols to an automated water and irrigation management system, as the object of this study, the MQTT protocol. The qualitative approach will be based on the development of a prototype, which will make it possible to validate the applicability of the project. The bibliographic research was carried out through an exploratory study on the concepts of automation, connectivity, democratization of technologies and conscious use of water, thus enabling the theoretical discussion of this work. From the analysis of the literature, it is possible to observe the importance of family farming in the Brazilian economy. One of its most interesting features is the little mechanization of the planting and harvesting processes, which is a great opportunity for the development of the modernization of these processes, such as the irrigation process, which is the reason for this project. In view of the results obtained, it is possible to observe the integration potential that the communication protocol provides to the irrigation system, showing efficiency in the exchange of information collected in the field and, still, offering support to applications that can be implemented in the future, such as water control. parameters inherent to the source code by exchanging information via MQTT.

Keywords: Irrigation, Supervisory System, MQTT Protocol.

SUMÁRIO

1. INTRODUÇÃO

Um dos principais motivos para o destaque mundial do Brasil na produção de commodities é o agronegócio (JANK, 2005). Pode-se dividir a atividade agrícola em dois grupos: agricultura moderna, onde se utiliza o processo de produção em larga escala, equipamentos modernos e a predominante contratação de funcionários; e agricultura familiar, que por outro lado, não dispõe de tantos recursos tecnológicos, e sua operação é concentrada nas mãos de um núcleo familiar.

O conceito de agricultura familiar é estabelecido pela Lei 11.326/2006. Para isso, é preciso que se detenha um terreno menor que quatro módulos fiscais, sendo que o valor do módulo fiscal no Brasil pode variar de 5 a 110 hectares, dependendo do município.

Segundo o IBGE, no ano de 2017, dentro da somatória de estabelecimentos agropecuários e aquicultores nacionais (5.073.324), 76,8% correspondiam à agricultura familiar. Sabe-se que o setor agrícola é o maior consumidor de água, somando a nível mundial, 69% de toda a água derivada das fontes (rios, lagos e aquíferos subterrâneos) e os outros 31% são consumidos pelas indústrias e uso doméstico (CHRISTOFIDIS, 1997). Como qualquer tipo de produção que carece de insumos para sua viabilização e desenvolvimento, a agricultura familiar se baseia essencialmente no uso e manejo de água.

1.1 Objetivos

Como mencionado anteriormente, a base de qualquer tipo de agricultura, seja ela mecanizada ou manual, moderna ou familiar, envolve o uso de água. Em algumas regiões, o fornecimento de água pode ser comprometido por diversos fatores, sejam eles de infraestrutura ou até mesmo de causas naturais. Dinâmicas sustentáveis como a criação de açudes ou reservatórios para coleta de água da chuva são essenciais; sendo elas uma excelente oportunidade para o desenvolvimento e implementação de tecnologias que auxiliem a integração das soluções ao cotidiano do produtor.

1.1.1 Objetivo geral

Desenvolver um sistema de irrigação integrado a um protocolo de comunicação em nuvem e acesso remoto.

1.1.2 Objetivos específicos

Para se tornar possível a implementação do protocolo de comunicação em nuvem, é necessário o desenvolvimento de um protótipo que simulasse as aplicações reais do sistema de irrigação, assim como o desenvolvimento da solução de conectividade ao protocolo. Com isso, tem-se os seguintes objetivos específicos:

- Escolha dos materiais utilizados;
- Desenvolvimento de código fonte para funcionalidade do sistema de irrigação;
- Integração do protocolo de comunicação MQTT ao código fonte;
- Desenvolvimento de um protótipo físico que simule o sistema de irrigação;
- Validação da integração do sistema físico com as aplicações propostas de automação.

1.2 Organização da monografia

O restante dessa monografia está organizado da seguinte maneira. O capítulo 2 aborda, de maneira explicativa, os conceitos teóricos correspondentes aos equipamentos utilizados no desenvolvimento do protótipo proposto. O capítulo 3 se refere à aplicação dos conceitos anteriormente citados, abordando a escolha dos materiais, desenvolvimento de *software*: métodos de desenvolvimento e de toda lógica de intertravamentos realizada, e métodos utilizados para a confecção do protótipo. Já no capítulo 4 é apresentado o relato dos resultados obtidos e comentários acerca do atingimento dos objetivos propostos. No capítulo 5 são apresentadas as conclusões da monografia. Por fim, no apêndice, constam informações complementares.

2. REVISÃO TEÓRICA

Neste capítulo é feita uma breve apresentação explicativa e resumida acerca dos conteúdos utilizados no desenvolvimento do projeto. Dentre eles, é possível visualizar a introdução do funcionamento do protocolo MQTT, que é a base de comunicação e responsável por transmitir os dados da plantação para o sistema de acesso remoto. Também é descrito o microcontrolador ESP8266, responsável por processar os dados dos sensores e realizar a comunicação por meio do protocolo MQTT. Além disso, é proposta uma revisão conceitual do sensor HC-SR04, utilizado para obtenção do nível de água no reservatório.

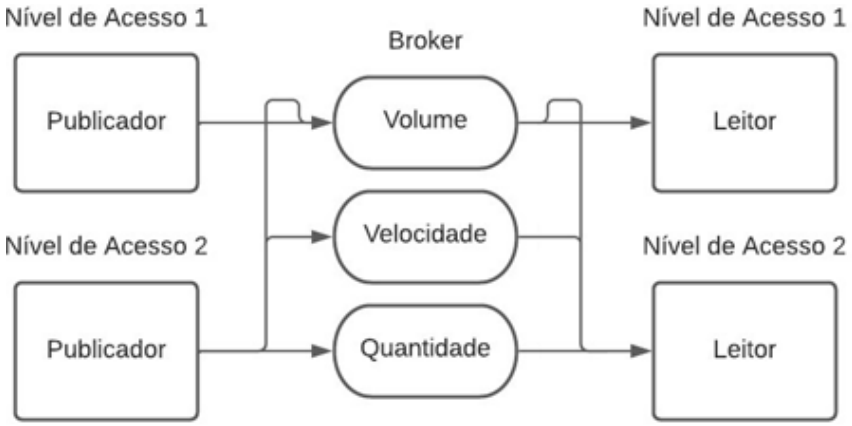
2.1 MQTT

Segundo Torres (2016), o MQTT (*Message Queue Telemetry Transport*) ou Transporte de Telemetria de Mensagens em Fila, criado pela *International Business Machines Corporation* (IBM), surgiu como solução à conectividade de sistemas de telemetria de oleodutos por satélite em 1999. Tratava-se de um protocolo máquina-máquina (M2M), que em 2013 foi padronizado por uma organização sem fins lucrativos, chamada OASIS, onde indivíduos, empresas e governos colaboraram na solução dos maiores problemas técnicos no desenvolvimento de códigos abertos. Essa organização é atualmente patrocinada por empresas como Dell, Oracle, Adobe e a própria IBM (OASIS, 2022). Outro fato importante a respeito do protocolo MQTT é que desde 2010 ele tem seu uso livre de *royalties*.

Com o passar do tempo, o protocolo foi amplamente utilizado em aplicações industriais, logísticas, automotivas e, inclusive, em internet das coisas (IoT). Segundo Torres (2016), o protocolo MQTT adota o protocolo TCP, como também o padrão de mensagem de publicador e assinante ou *Publisher/Subscriber*. As partes que compõem a rede de comunicação podem ser cadastradas como: “publicadores”, podendo assim emitir dados à rede; “assinantes”, que possuem apenas capacidade de leitura de dados; e

“publicadores/assinantes”, onde os dispositivos que acessam a rede podem tanto realizar a leitura quanto emitir dados.

Figura 1: Níveis de Acesso



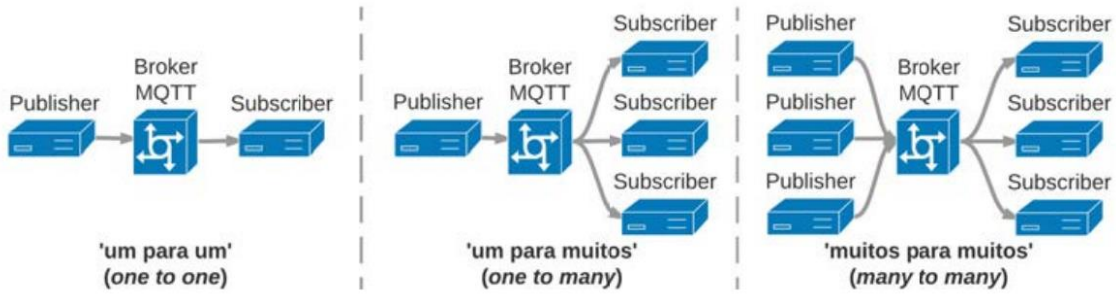
Fonte: Autores

Adicionalmente, define-se o nível de acesso destinado a cada dispositivo conectado, como mostrado no exemplo da Figura 1. Fica evidente a relação descrita acima, onde os publicadores de nível de acesso 1 poderão apenas enviar dados para o tópico de volume; já o publicador de nível 2 poderá realizar o envio de dados para todos os tópicos. A mesma regra se aplica para os leitores, onde o nível de acesso 1 consegue ler os dados apenas de volume; enquanto o nível de acesso 2 tem acesso aos dados de volume, velocidade e quantidade.

A estrutura do MQTT se baseia na utilização de um agente intermediador, chamado de Broker. Esse agente é responsável por fazer o tráfego e armazenar os dados; dessa forma, ao realizar uma publicação, os dados armazenados no broker são atualizados tanto durante o acesso do leitor ou em um novo acesso.

Conforme Torres (2016), o tráfego de informações pode ocorrer em três formas, como *one-to-one*, *one-to-many* e *many-to-many*. Todos esses casos podem ser visualizados no esquema da Figura 2.

Figura 2: Distribuições de dados MQTT



Fonte: Torres (2016)

2.2 ESP8266 – NodeMCU

O ESP8266 é um dispositivo microcontrolado, produzido pela *Espressif Systems*, empresa multinacional chinesa com sede em diversos países, sendo um deles o Brasil.

Figura 3: ESP8266

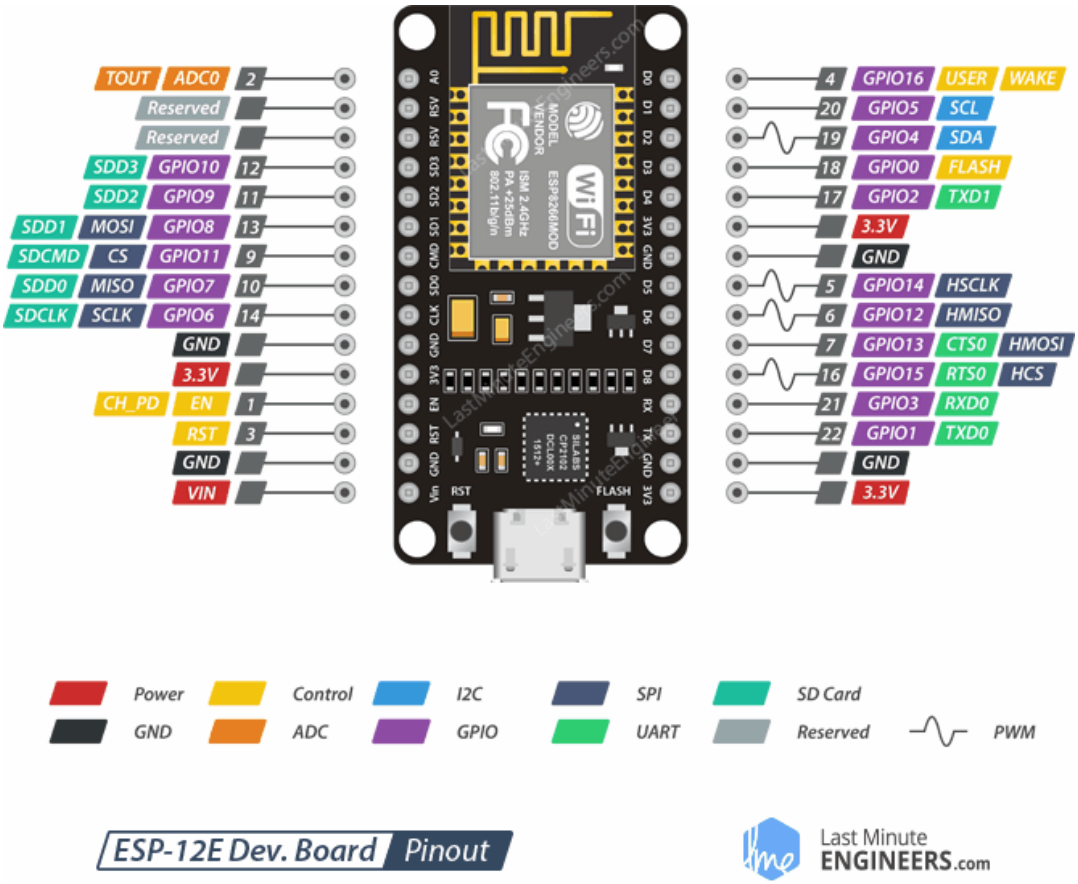
Fonte: Byteflop

O dispositivo está no mercado desde 2014, e atualmente é extremamente difundido. Seu grande diferencial, além do preço que varia de R$20,00 a R$50,00, é a

capacidade de conexão sem fio, devido ao módulo *wi-fi* acoplado em sua placa. Por conta disso, sua aplicação passa a ser extremamente variada, podendo ser utilizado como módulo *wi-fi* por outros microcontroladores ou até mesmo para desenvolver aplicações embarcadas que utilizem somente o ESP8266 (OLIVEIRA, 2017).

A família de dispositivos *DevKits* ESP comercializados, compartilha de um processador de 32 Bits com 2,4 GHz, havendo mudanças em suas interfaces que resultam em mais ou menos entradas, proporcionando versatilidade de uso dos dispositivos. Ao se considerar as proporções minimizadas de prototipagem do projeto sugerido, foi possível recorrer a uma das plataformas mais básicas do *DevKit*, mostrado na Figura 4.

Figura 4: ESP8266 Pinagem



Fonte: GitLab UFRGS

Para completar e ampliar as possibilidades de aplicação do ESP8266, o dispositivo utiliza um *firmware* chamado NodeMCU. Esse *firmware* nada mais é que uma plataforma

open source de IoT, baseada no próprio ESP8266. Nela, é possível realizar a programação do dispositivo utilizando a IDE do Arduino, graças à porta USB presente no módulo.

O desenvolvimento de código para a plataforma pode ocorrer com a utilização de diversas ferramentas e métodos de criação. Dentre eles, temos o controle GPIO, HTTP *request* HTTP *server,* conectando a um MQTT *broker*, dentre outras aplicações.

2.3 HC-SR04

O dispositivo HC-SR04 é um sensor ultrassônico produzido pela empresa Elecfreaks. O dispositivo consegue comportar, mesmo de forma compacta, um circuito de controle, um transmissor e um receptor ultrassônico. Sua precisão fornece medidas na casa de 20mm a 4000mm, com uma precisão de 3mm para os valores medidos, (NAKATANI, 2014).

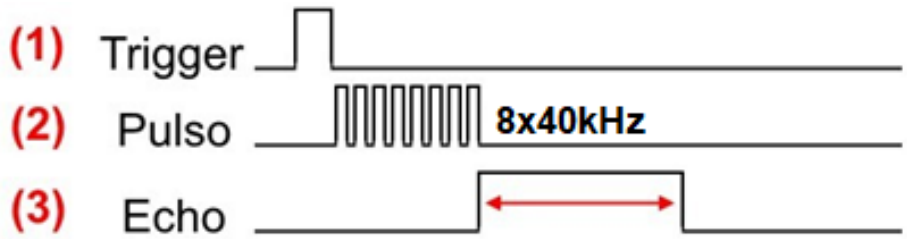
Figura 5: Sensor HC-SR04

Fonte: FilipeFlop

O funcionamento do componente ocorre da seguinte maneira: após receber um pulso de 5V por um tempo mínimo de 10µs, o sensor inicia um processo de leitura de distância e utiliza o transmissor para enviar uma série de 8 ciclos de pulso ultrassônico com 40kHz de frequência base. Logo em seguida, envia um sinal positivo no terminal

“ECHO”, que permanece no estado positivo até o sensor receptor detectar o sinal dos pulsos enviados, o que normaliza o ponto “ECHO” em nível lógico 0, como mostrado na Figura 6.

Figura 6: Diagrama de tempo HC-SR04



Fonte: Blog Filipeflop

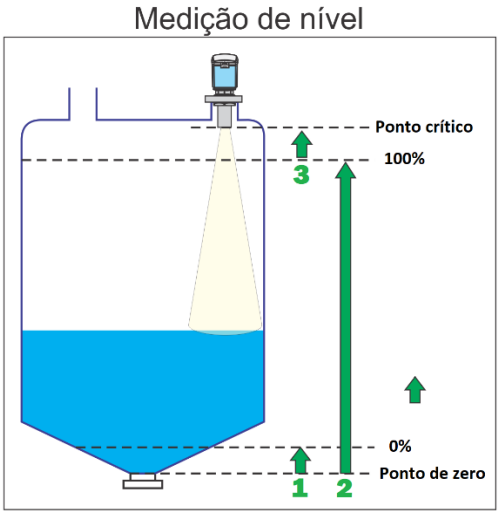
Essa dinâmica possibilita o desenvolvimento do cálculo de distância, aproveitando o parâmetro de tempo medido pelo sensor. A equação utilizada para o cálculo de distância, conforme mostrado anteriormente, pode ser resumida pela equação (1):

$$Lc = \frac{Tempo\ ECHO\ em\ High\ \times\ Velocidade\ do\ Som}{2} \quad (1)$$

Na equação (1), a “Velocidade do Som” é considerada, pois os sinais dos pulsos ultrassônicos são transportados em uma velocidade equivalente à do som e, por conta disso, o valor que deverá ser utilizado é de 340 m/s. Já o “Tempo ECHO em High” é o sinal do sensor HC-SR04, resultado do tempo necessário para que o sinal enviado seja recebido pelo sensor do HC-SR04 (envio e recepção), fazendo com que seja necessário dividir o valor medido por dois para que se alcance somente o valor da distância entre o sensor e o objeto testado. O cálculo necessário pode ser realizado em qualquer dispositivo microcontrolado, como o Arduino e o próprio NodeMCU.

O sensor foi instalado no topo do nosso reservatório, de modo que, ao realizar a leitura da distância entre o sensor e a superficie da aguá no reservatório, é possível obter o atual nível de água dentro do reservatório, conforme ilustrado na Figura 7.

Figura 7: Exemplo de Leitura de Nível de Água



Fonte: Autores

O sistema foi construído para que o nível de água nunca alcance o ponto crítico ou o ponto de zero, realizando a leitura e controle de nível de água entre de 0 e 100%.

2.4 Sensor Capacitivo de umidade do solo

O sensor de umidade capacitivo do solo é uma das melhores soluções para realizar a medição do solo, pois o seu design de construção ajuda a evitar o principal problema que existe em um sensor de umidade do solo comum (resistivo), isto é, a corrosão em seus contatos.

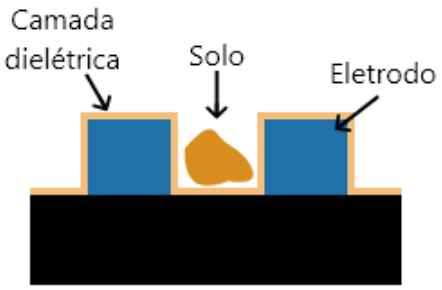
Figura 8: Sensor Capacitivo de umidade do solo

Fonte: Curtocircuito

O funcionamento do sensor se baseia, como o próprio nome diz, na alteração da capacitância a partir da umidade do solo. Em sua placa, existem dois eletrodos que ficam distantes um do outro. É possível visualizar esses dois eletrodos no sensor, observando um relevo que existe na superfície da placa.

Na Figura 9, é possível observar a camada de material dielétrico que os eletrodos possuem. Essa camada é responsável pela formação de um capacitor a partir dos eletrodos, bem como sua proteção à corrosão.

Figura 9: Ilustração do funcionamento do sensor a partir do solo



Fonte: Mundoprojetado

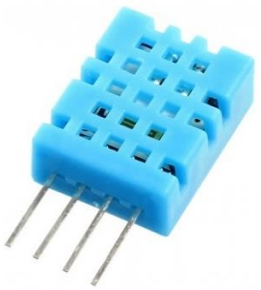
Conforme o solo vai preenchendo o espaço entre os eletrodos, altera o meio do capacitor e o campo elétrico, alterando então a sua capacitância. A variação de umidade

do solo vai provocar variação no campo elétrico entre os eletrodos, provocando a variação de capacitância. Logo, obtemos os valores da umidade do solo a partir da capacitância.

2.5 Sensor DHT11

O DHT11 é um sensor que permite realizar a leitura da temperatura umidade do ambiente. Esse possui a capacidade de realizar leituras de temperatura entre 0ºC e 50ºC, e umidade relativa do ar entre 20 e 90%.

Figura 10: Sensor DHT11



Fonte: FilipeFlop

Este sensor foi implementado para observar a temperatura e umidade do ambiente e permitir futuras atualizações no projeto.

2.6 Bomba de Água

A bomba de água selecionada para o projeto é uma bomba muito utilizada em aquários domésticos. Ela possui parâmetros uma alimentação de 12VDC, potência de 19W e uma vazão de 800L/h. Possui fácil conexão com mangueiras, descartando a necessidade de adaptadores. O seu sistema de funcionamento interno não possui escovas, garantido a isolação entre a água e o sistema elétrico da bomba.

Figura 11: Bomba de Água

Fonte: Americanas

2.7 Válvula Solenóide de água

A válvula solenoide de água selecionada para o projeto é uma válvula muito utilizada em lavadoras automáticas. Ela possui como parâmetros uma alimentação de 110VAC, vazão mínima de 7L/min e vazão máxima de 40L/min. Possui rosca externa de 3/4", facilitando a conexão com o sistema doméstico de torneiras.

Figura 12: Válvula Solenóide de Água

Fonte: Mercado Livre

Esse modelo de válvula foi selecionado por suas características de alimentação e vazão, pois em projetos futuros, será capaz de suportar um sistema hidráulico mais robusto.

3. DESENVOLVIMENTO

Para que fosse possível validar as características teóricas e técnicas que envolvem a sua construção, foi realizado o processo prático do protótipo. O desenvolvimento se deu a partir da escolha dos materiais, código fonte e confecção do protótipo. O desenvolvimento e a implementação foram realizados conforme as seguintes etapas:

- Seleção e definição dos dispositivos utilizados;
- Desenvolvimento do código fonte e integração do protocolo de comunicação MQTT;
- Desenvolvimento do protótipo e simulação prática do sistema de irrigação.

3.1 Escolha dos materiais

O processo de escolha dos materiais prosseguiu com a premissa de utilização de materiais e tecnologias que fossem familiares, de maneira a facilitar a integração ao projeto; e dispositivos encontrados com fácil acesso no mercado e que não houvesse a necessidade de realizar importação dos equipamentos.

3.1.1 Fonte Chaveada 12V

Para realizar conversão de corrente alternada para corrente continua e alimentar todo circuito eletrônico do projeto, foi necessário utilizar uma fonte chaveada bivolt que transforma a tensão de 100-240V de corrente alternada para 12V de corrente contínua. Ela é responsável por suprir a tensão necessária para alimentar a fonte integrada do circuito, bomba de água e os reles.

3.1.2 ESP8266

Um dos aspectos mais importantes do projeto é o acesso sem fio e remoto aos dados coletados sobre as informações pertinentes ao plantio, como o volume de água nos reservatórios e a certificação da umidade geral do solo regado. Para isso, seria necessário possuir um dispositivo com conexão sem fio à internet e que, além disso, tivesse fácil integração com o protocolo MQTT, que é base de toda troca de informações na rede. Com base nesses requisitos, adotamos o ESP8266.

Ao analisar os dispositivos presentes no mercado, é fácil considerar a utilização do dispositivo, pois como mencionado anteriormente, o dispositivo possui um módulo de conexão Wi-Fi, além de possuir as bibliotecas necessárias, com base no Arduino, para implementação do protocolo MQTT. Sendo assim, responsável pelo controle das devidas atuações do sistema, é responsável também pelo gerenciamento das informações enviadas ao *broker* MQTT.

3.1.3 LM7805

Durante o desenvolvimento do projeto, utilizamos uma fonte 12V para alimentar o todo o circuito eletrônico, mas a maioria dos microcontroladores funcionam com tensões abaixo de 12V. Logo, era necessário utilizar um bom regulador de tensão de 12V para 5V para criar uma fonte integrada à placa eletrônica e alimentar o projeto.

O regulador escolhido foi o LM7805, um regulador amplamente utilizado na eletrônica que atende as necessidades do projeto, podendo operar em sua entrada com níveis de tensão de 7 a 20V, garantindo uma saída, dentro de seus parâmetros de utilização, de 5V e 1A.

3.1.4 TIP127

Após ser determinado o uso do regulador de tensão, foi proposto a utilização de outro dispositivo, o TIP127, com o objetivo de garantir que o LM7805 não trabalhasse

no limite de fornecimento de corrente; permitindo que o LM7805 mantivesse a tensão em 5V e o TIP127 fosse encarregado de fornecer toda a corrente para o circuito, tendo como limite máximo de fornecimento de corrente de 5A.

3.1.5 BC547

O sinal de saída de um microcontrolador é de baixa corrente. Sozinho, ele não tem a capacidade de alimentar um relé. A solução foi utilizar o BC547 como chave eletrônica. No projeto, seu funcionamento consiste em permitir a passagem de corrente entre coletor e emissor ao receber um sinal do microcontrolador em sua base. Dessa forma, é possível utilizar ele como chave eletrônica para permitir a passagem de 12V nos relés e nos leds.

3.1.6 Relé Eletromecânico

Para realizar o acionamento das cargas do projeto, como as válvulas e a bomba, selecionamos relés eletromecânicos de 12V para atuar como chaves para o acionamento das cargas. As principais motivações para selecionarmos os relés, está suas características elétricas e baixo custo, pois ao utilizar eles como chave, é possível criar uma isolação entre o circuito de controle e as cargas, e suportar com facilidade a cargas propostas no projeto.

3.1.7 Sensor capacitivo de umidade do solo PH2.0-3P

Inicialmente, foi proposta a utilização de um sensor de umidade resistivo, porém o modelo que foi utilizado apresentou algumas dificuldades de implementação, como a imprecisão de leitura dos sinais analógicos de umidade e a sensibilidade à corrosão. Fatores que foram importantes na escolha do modelo capacitivo, uma vez que não apresenta os empecilhos do resistivo, sua configuração é adequada ao projeto, possuindo uma tensão de entrada de 3,3V a 5.5 V.

3.2 Desenvolvimento de Software

O desenvolvimento de *software* foi realizado utilizando a plataforma Arduino de programação, pois o mesmo é a base de programação do dispositivo microcontrolado utilizado no projeto, o NodeMCU. Outra característica muito importante é a facilidade na implementação dos dispositivos sensores como o HC-SR04 e o DHT11 pelo uso das bibliotecas presentes na aplicação de desenvolvimento.

A lógica desenvolvida, como pode ser observado no APÊNDICE**Erro! Fonte de referência não encontrada.**, é baseada na separação em três blocos principais: utilização de água, conectividade e irrigação.

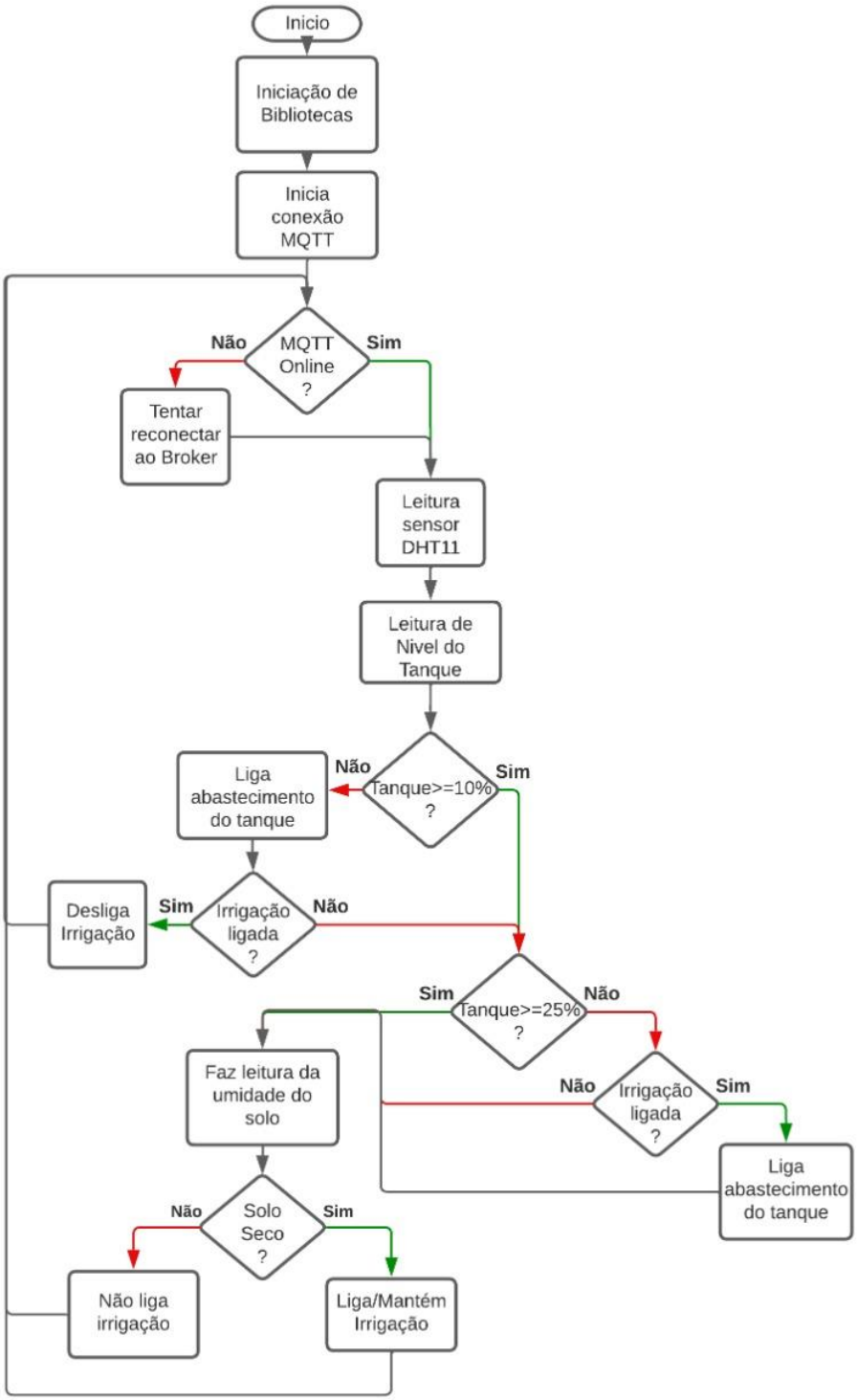
Para o bloco de "utilização de água", foi desenvolvida a seguinte lógica: a principal fonte a ser utilizada no processo de irrigação é a água proveniente da captação das chuvas, sendo que quando for alcançado um nível crítico, será necessário o suporte da fonte de água potável da rede de abastecimento. Uma vez que o tanque for abastecido até seu nível de "segurança", a fonte da rede de abastecimento deverá ser fechada, voltando a ser utilizada a água abastecida no reservatório. Porém, uma vez que o reservatório alcance o nível crítico e a fonte de abastecimento não seja o suficiente para normalizar nível do reservatório, o processo de irrigação será encerrado até o nível de "segurança" ser alcançado novamente, a fim de preservar o conjunto motor.

Já para "conectividade", a lógica foi desenvolvida da seguinte maneira: todas as informações captadas pelos sensores de umidade do solo são enviadas ao sistema de troca de informações, o *broker*, podendo ser visualizadas por qualquer dispositivo conectado a ele como, até mesmo, o telefone celular.

Por último, para o sistema de "irrigação" o desenvolvimento ocorre da seguinte maneira: o processo de irrigação irá captar as informações de umidade do solo e, por um processo de comparação, irá atuar os conjuntos de bombeamento de água. Os parâmetros utilizados para efetuar a comparação poderão ser alterados previamente, utilizando a conexão via protocolo MQTT.

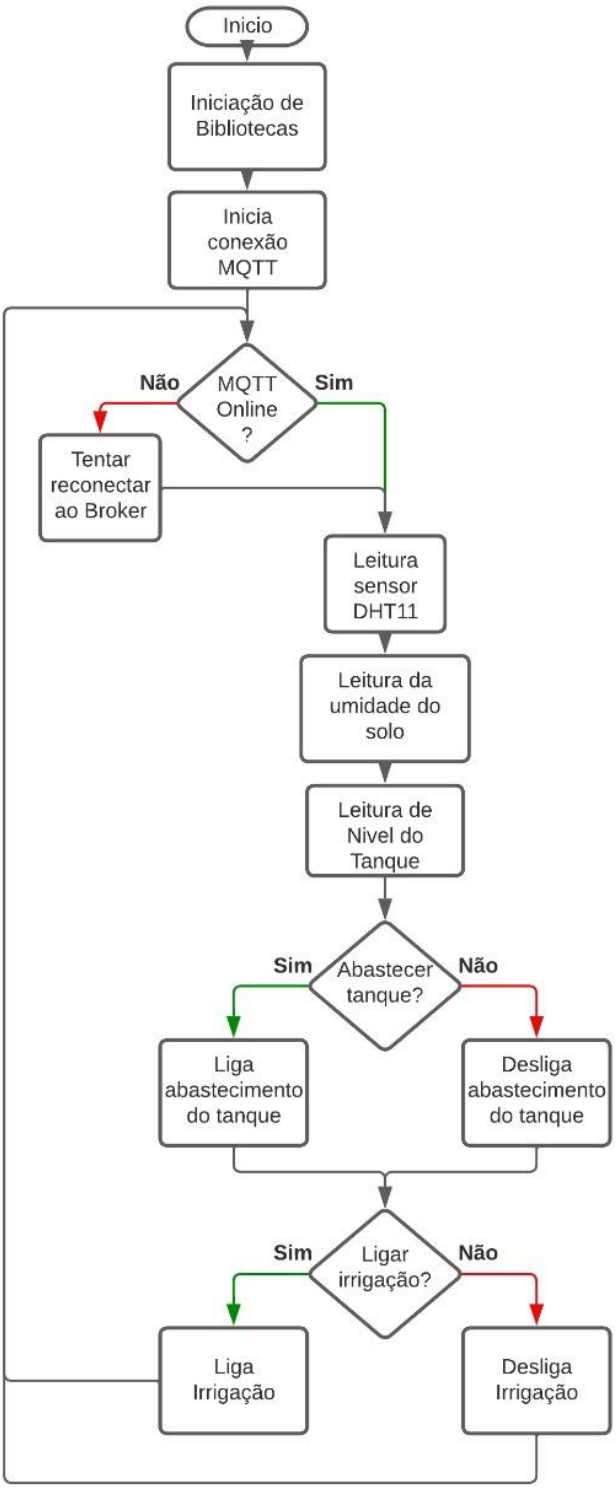
O código fonte foi desenvolvido a partir das premissas descritas anteriormente, levando em consideração a ativações automáticas e manuais, com base nos fluxogramas a seguir, Figura 13 e Figura 14.

Figura 13: Fluxograma lógico modo automático



Fonte: Autores

Figura 14: Fluxograma lógico modo manual



Fonte: Autores

3.3 Aplicação

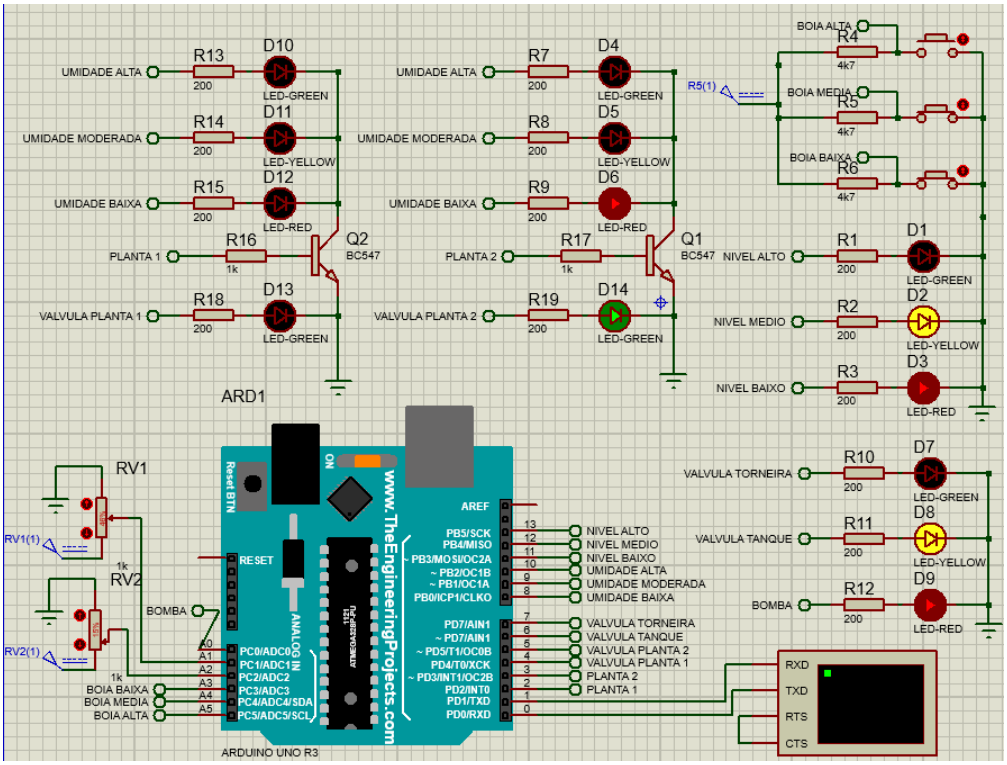
A aplicação foi completamente desenvolvida no *software* Proteus, onde foi possível realizar testes prévios de simulação que facilitaram no processo de confecção do protótipo físico. Logo após o projeto via *software*, foi desenvolvido o processo de confecção do circuito impresso à placa de fenolite, utilizando para isso o processo de transferência térmica de imagem.

3.3.1 Projeto e testagem

O processo de validação lógica do projeto foi iniciado após a definição dos componentes que seriam utilizados. Como o *software* utilizado proporciona a possibilidade do desenvolvimento de simulações, inclusive de sistemas microcontrolados, foi necessário realizar prévio desenvolvimento do código fonte. O *software* utilizado para realizar as simulações práticas foi o Proteus 8.9.

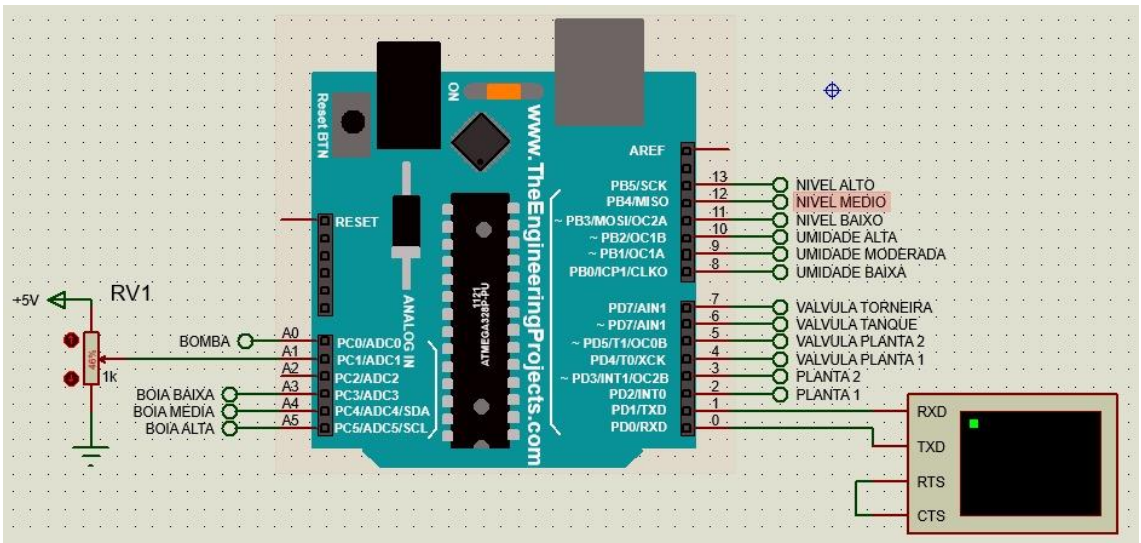
Para viabilizar a capacidade de simulação dos dispositivos microcontrolados no *software* utilizado, foi necessário implementar ao projeto as respectivas bibliotecas de cada componente. Pelo fato das bibliotecas de NodeMCU não terem sido encontradas, a dupla optou por utilizar a biblioteca do próprio ArduinoUNO. Como pode ser visualizado na Figura 16.

Figura 15: Protótipo inicial de irrigação



Fonte: Autores

Figura 16: Protótipo inicial - Detalhe Arduino

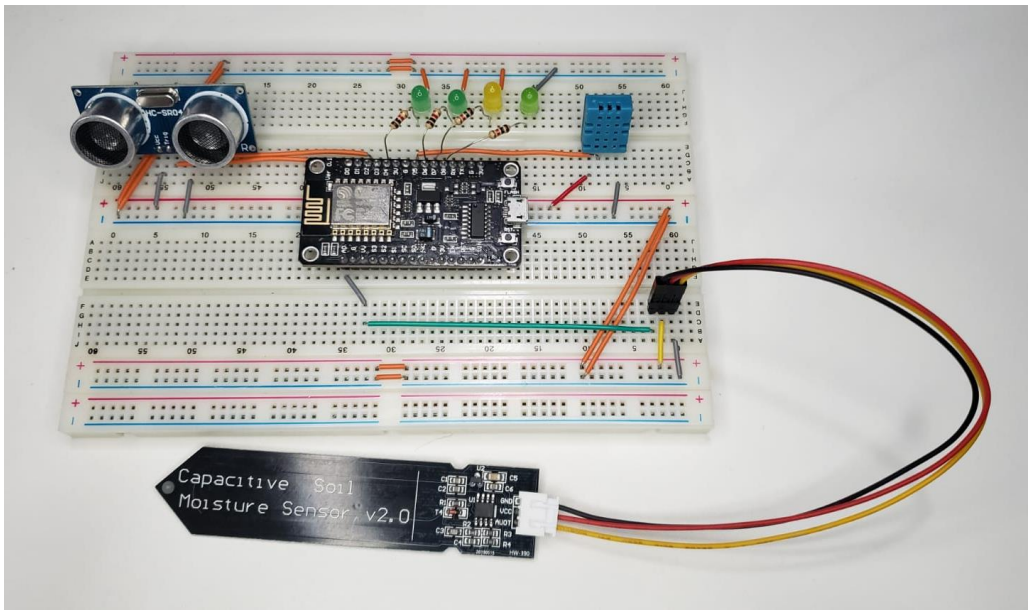


Fonte: Autores

A utilização se torna efetiva como parâmetro de validação lógica, pois o código fonte utilizado na simulação pode ser facilmente reaproveitado no desenvolvimento posterior do NodeMCU, justamente por compartilharem da plataforma Arduino.

Uma vez que a base de aplicação foi validada via *software*, foi dado início ao desenvolvimento do *hardware* do protótipo. Esta etapa foi constituída na aplicação dos conceitos validados via *software* em uma *Protoboard*. A montagem da mesma pode ser observada na Figura 17.

Figura 17: Montagem de protótipo

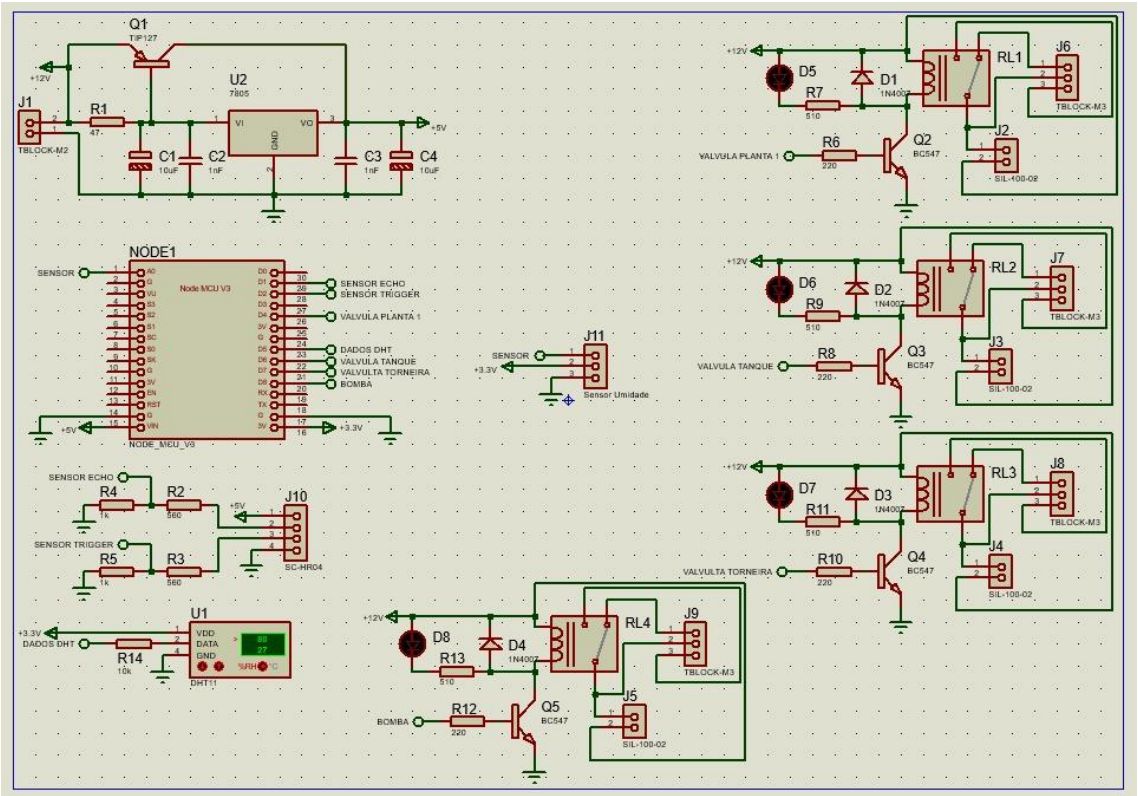


Fonte: Autores

Durante a utilização da *Protoboard*, foi possível comprovar o funcionamento do dispositivo NodeMCU, utilizando o código fonte desenvolvido com base em Arduino, assim como a conexão à internet e a aplicação MQTT.

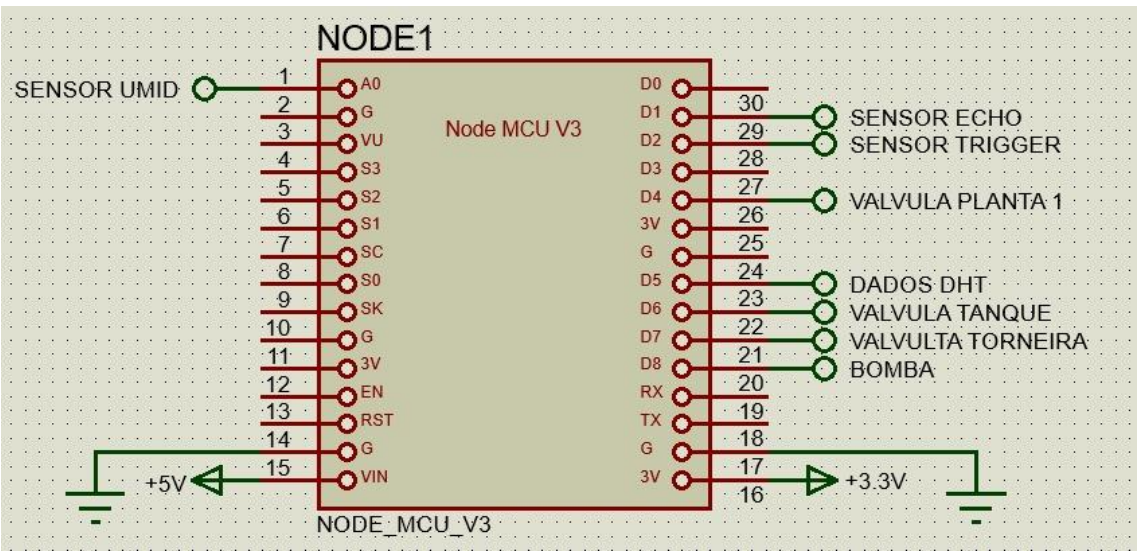
Após a validação da aplicação, tanto de maneira prática via *Protoboard* quanto sua simulação proporcionada pelo *software* Proteus, foi dado início ao processo de confecção do modelo PCB. Para isso, foi necessário desenvolver um novo arranjo esquemático dos componentes para adequar as conexões do NodeMCU.

Figura 18: Projeto final com NodeMCU



Fonte: Autores

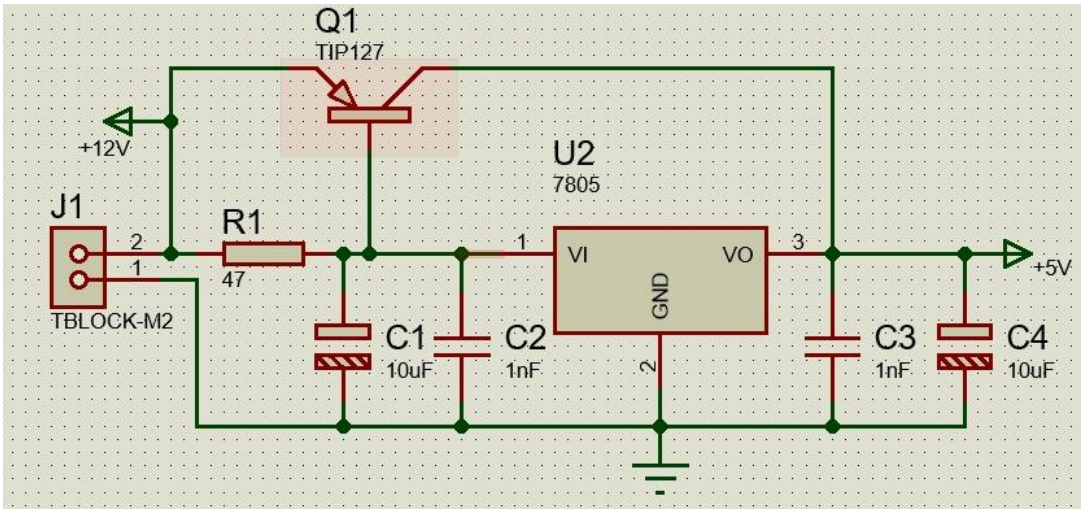
Figura 19: Projeto final - Detalhe NodeMCU



Fonte: Autores

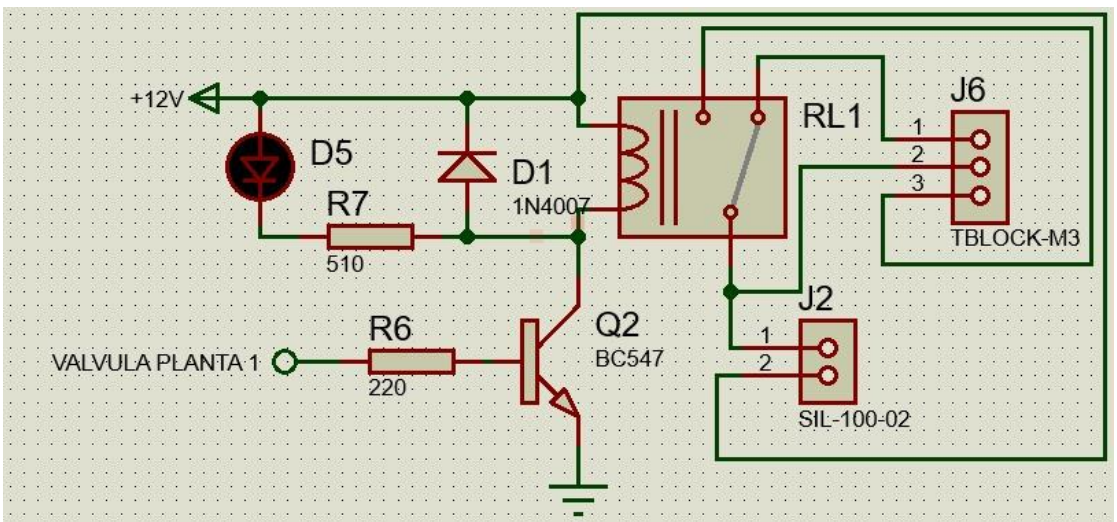
Na Figura 18 é possível observar o esquema elétrico criado no software Proteus com o objetivo de iniciar a confecção de um modelo PCB. Neste esquema é possível observar quais pinos do microcontrolador serão utilizados e quais as conexões de todos os componentes da PCB. Para melhorar a visualização, foi separado em blocos cada parte do projeto, ilustrados nas Figura 19 a Figura 23.

Figura 20: Projeto final - Entrada de energia



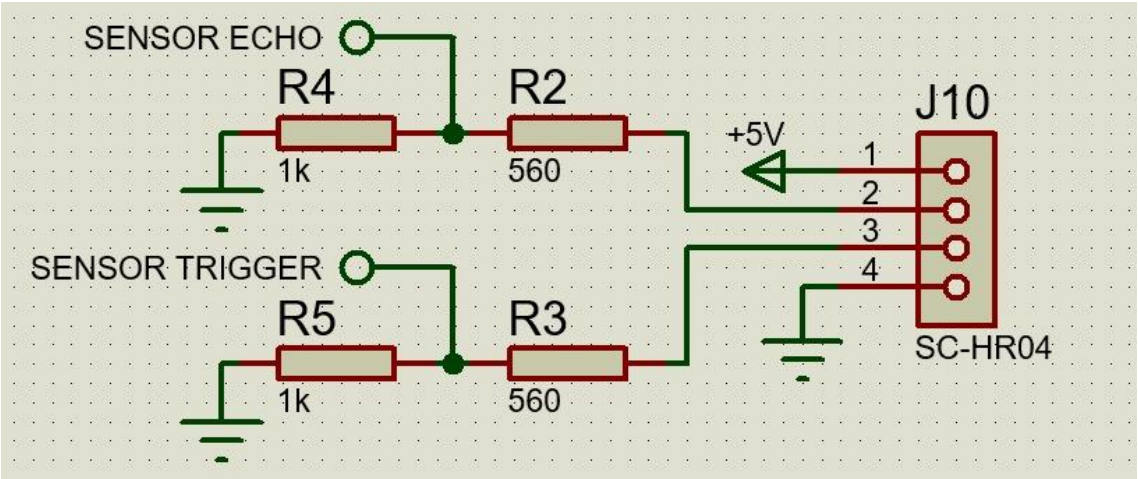
Fonte: Autores

Figura 21: Projeto final - Arranjo utilizando para acionamento dos relés



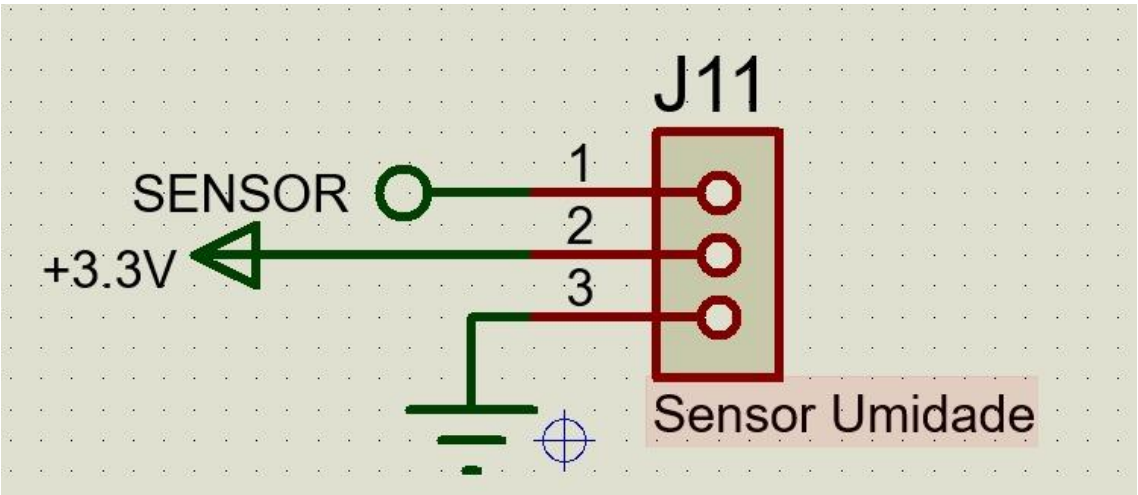
Fonte: Autores

Figura 22: Projeto final - Conexão SC-HR04



Fonte: Autores

Figura 23: Revisão final - Conexão sensor de umidade

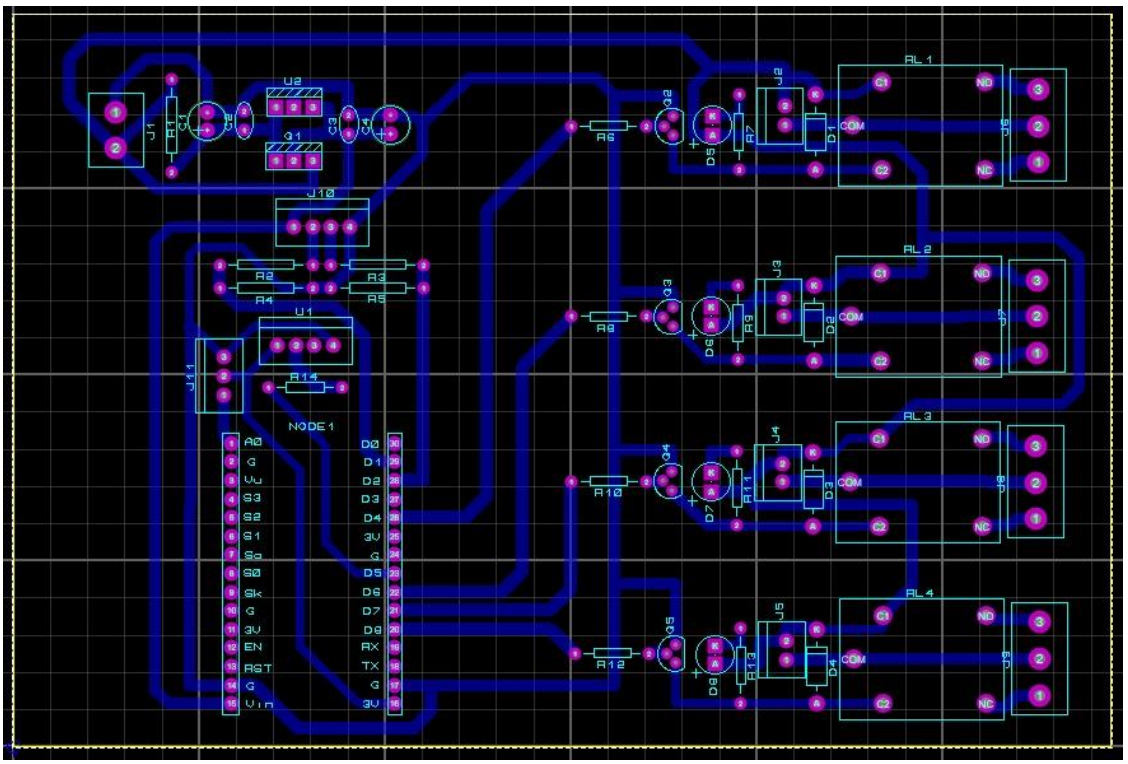


Fonte: Autores

O *layout* completo da placa foi desenvolvido com o auxílio das ferramentas presentes no Proteus. O processo ocorreu da seguinte maneira: definição da geometria e das dimensões da placa que seria utilizada; disposição manual dos componentes; e roteamento das conexões entre componentes. Para isso, foi utilizado processo de roteamento automático, porém mesmo utilizado a ferramenta do Proteus, foi necessário realizar correções de curso das trilhas de maneira manual.

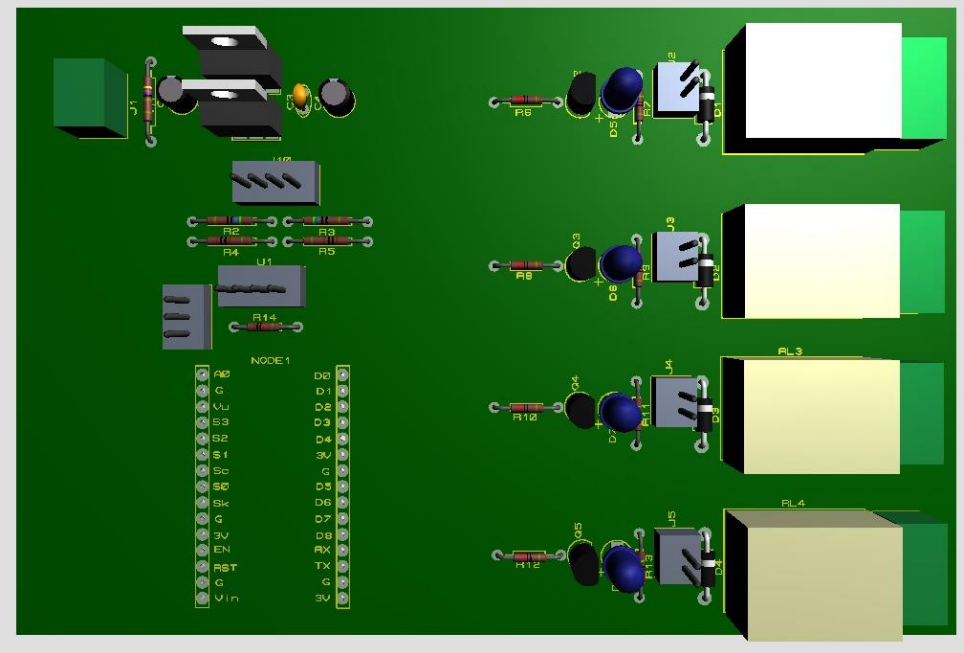
Os resultados da disposição dos componentes e conexões das trilhas que conectam os diversos dispositivos podem ser visualizados na Figura 24. Em sequência, a simulação 3D da placa de circuito impresso com os componentes já inseridos nas figuras 25 e 26.

Figura 24: PCB Layout Manual Placement e Auto-router.



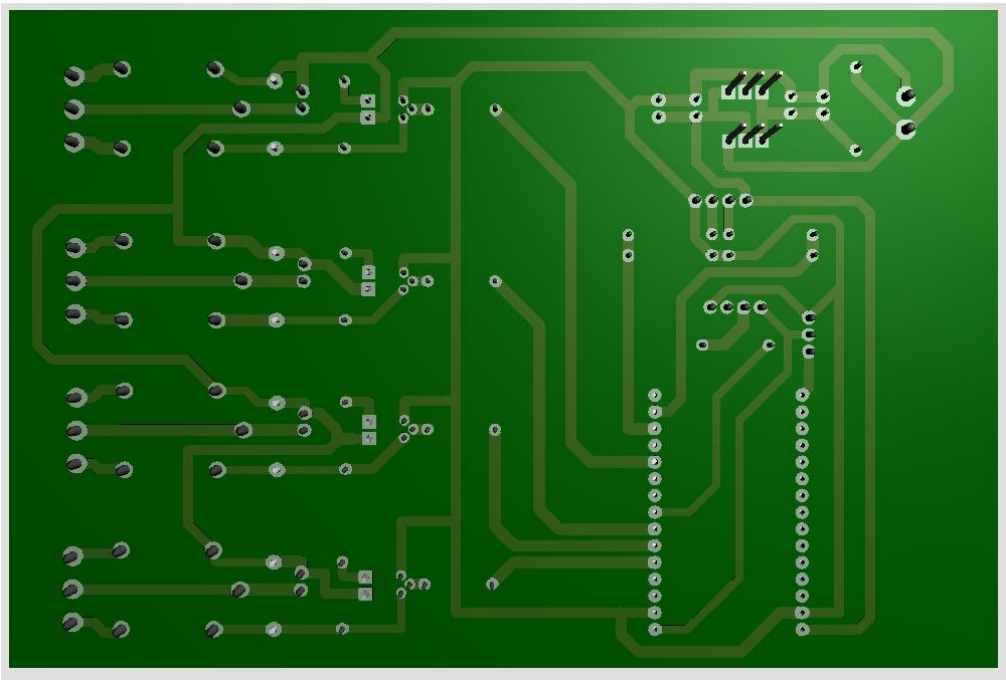
Fonte: Autores.

Figura 25: Visão frontal PCB



Fonte: Autores

Figura 26: Visão traseira PCB



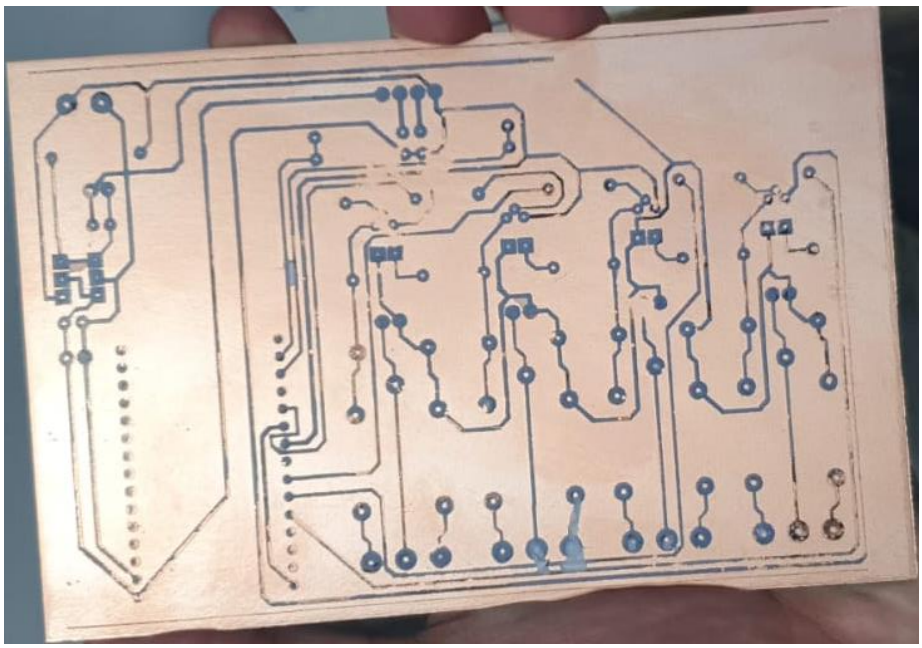
Fonte: Autores

3.3.2 Confecção da PCB

O processo escolhido para confecção da PCB foi o de transferência térmica, em que o *layout* desenvolvido é impresso e, em seguida, transferido para o lado cobreado da placa que será utilizada. Diante disso, é possível ressaltar alguns aspectos importantes para o sucesso do método. A impressora necessária para realizar a impressão deve ser a laser e o papel utilizado deve ser do tipo fotográfico, com acabamento *glossy*.

Durante as primeiras tentativas de transferência térmica, nem todas as condições foram atendidas, o que resultou em algumas falhas na transferência para a PCB, onde seria necessário realizar muitas correções com caneta para que fosse possível garantir que o percloreto fizesse corretamente o processo de corrosão do cobre. Foram realizadas uma série de quatro testes, e em todos foi possível observar o mesmo padrão de falhas na transferência, como visto na Figura 27.

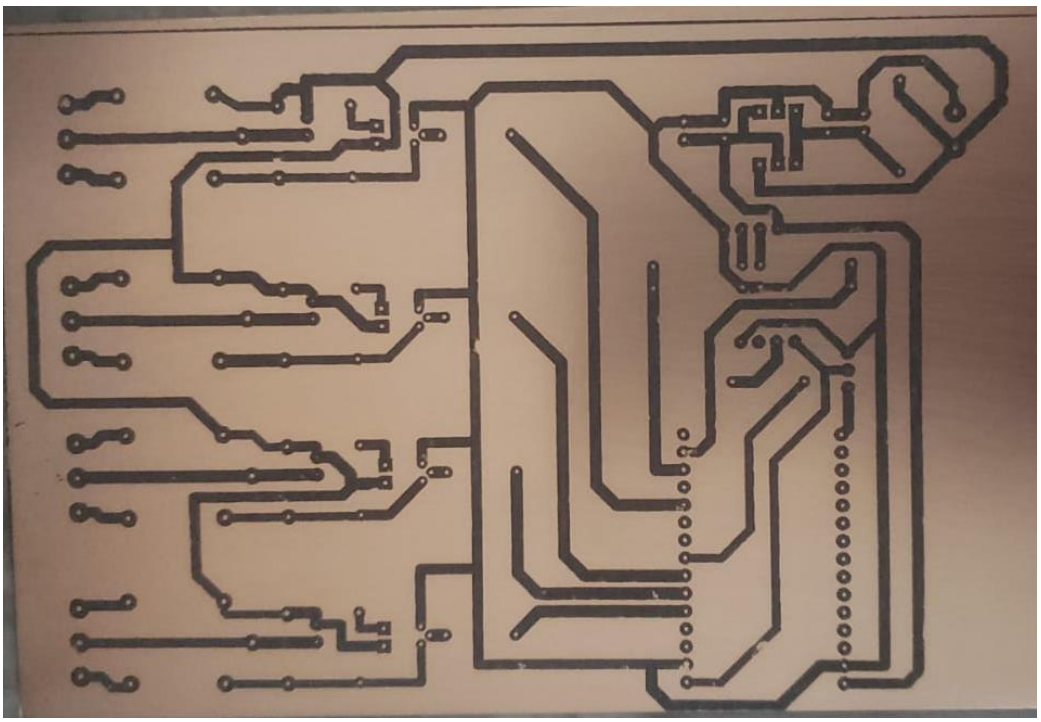
Figura 27: Falha na transferência térmica - Layout teste



Fonte: Autores

Após os diversos testes mencionados, foi observado que o papel antes utilizado, apesar de ser fotográfico e possuir acabamento *glossy*, não era próprio para impressoras a laser, e sim para as de jato de tinta. Logo após a troca do papel, foi possível obter um resultado extremamente satisfatório.

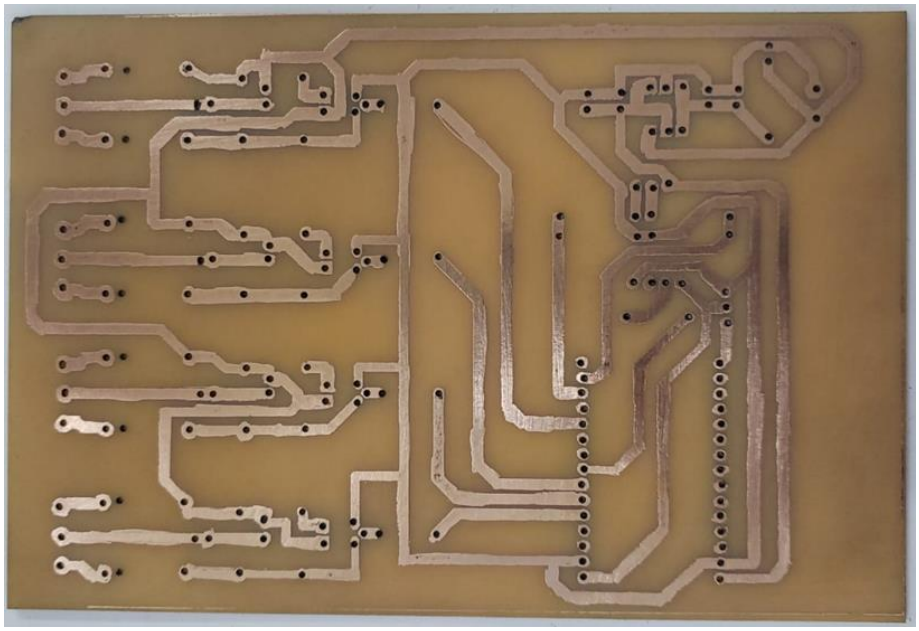
Figura 28: Transferência térmica com folha adequada - Layout Final



Fonte: Autores

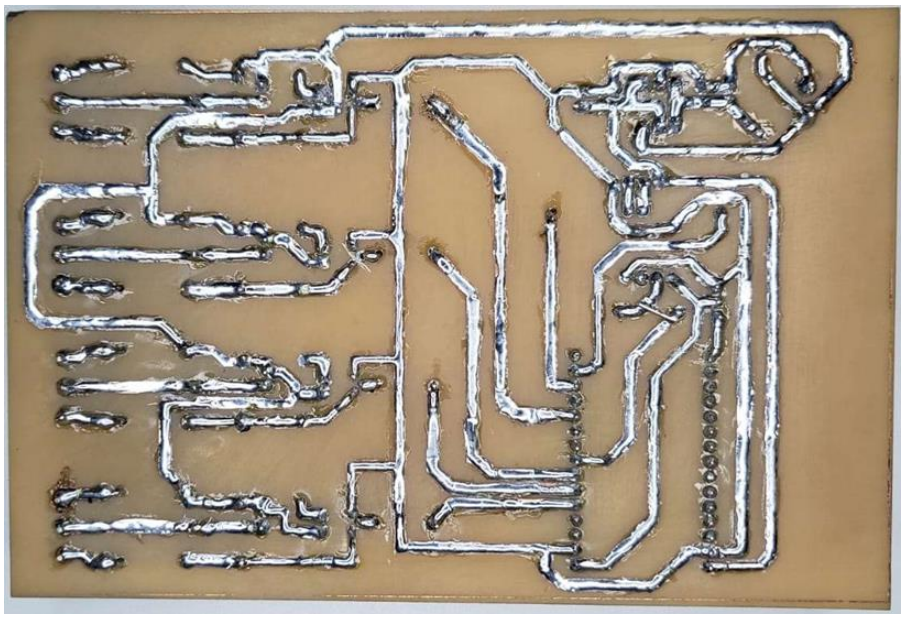
Como é possível observar na Figura 28, houve um salto notável de qualidade de impressão, sendo necessário realizar apenas algumas revisões a fim de garantir a correta corrosão por percloreto. Após a primeira leva de testes, a disposição dos componentes na placa foi modificada, alterando o visual do layout. A mudança ocorreu, pois caso fosse necessária a correção de possíveis falhas, não haveria dificuldade na utilização da caneta corretora. Após a conclusão do processo de transferência térmica, foi possível realizar a corrosão do cobre descoberto pela transferência térmica. O resultado pode ser observado naFigura 29. Após a corrosão, foi possível soldar as trilhas e os componentes que foram dispostos na placa. O resultado pode ser visualizado na Figura 30 e na Figura 31.

Figura 29: Resultado do processo de corrosão do cobre e furação da PCI



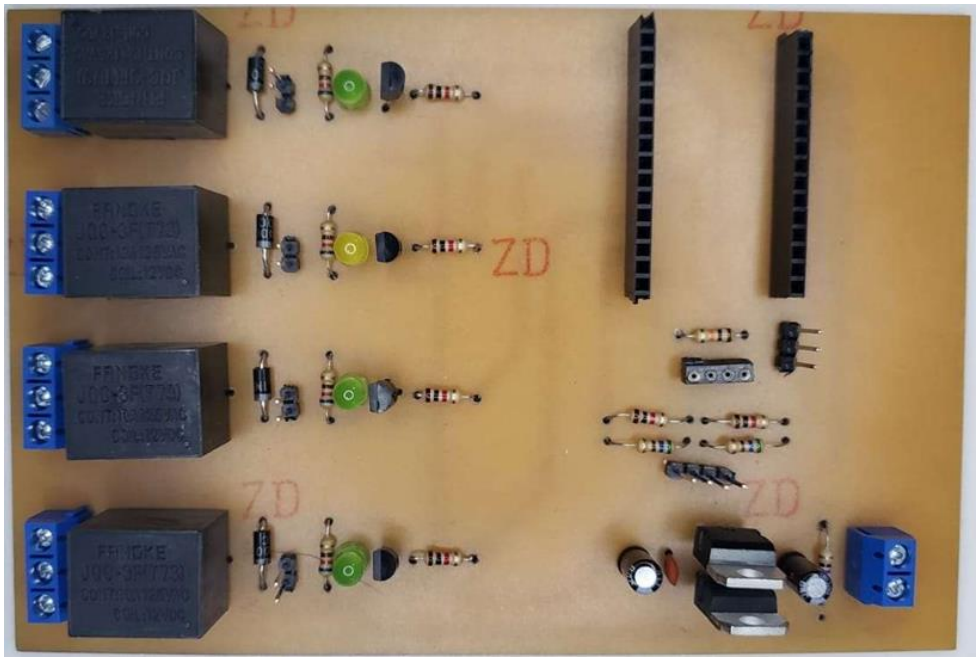
Fonte: Autores

Figura 30: Trilhas cobertas por estanho e componentes fixados - Face inferior



Fonte: Autores

Figura 31: Componentes fixados - Face superior



Fonte: Autores

3.3.3 Integração com protocolo de comunicação MQTT

Para complementar a implementação da integração ao protocolo foi necessário o desenvolvimento de uma interface homem máquina para que o usuário pudesse ter acesso às informações presentes no *broker* MQTT. Para isso foi utilizado uma plataforma gratuita de integração ao *broker*, "MQTT Dash". Nela é possível realizar a integração utilizando as ferramentas e modelos presentes na plataforma, assim como a configuração das permissões dos acessos que serão realizados ao *broker*.

É possível observar, na Figura 32, a utilização da plataforma, onde foi desenvolvida uma interface em que o usuário pudesse ativar o sistema e que pudesse ter acesso às informações coletadas pelos sensores conectados. Informações como temperaturas, umidade do solo presentes na Figura 32 e nível do tanque, presente na Figura 33.

Figura 32: Integração MQTT Dash 1/2

Fonte: Autores

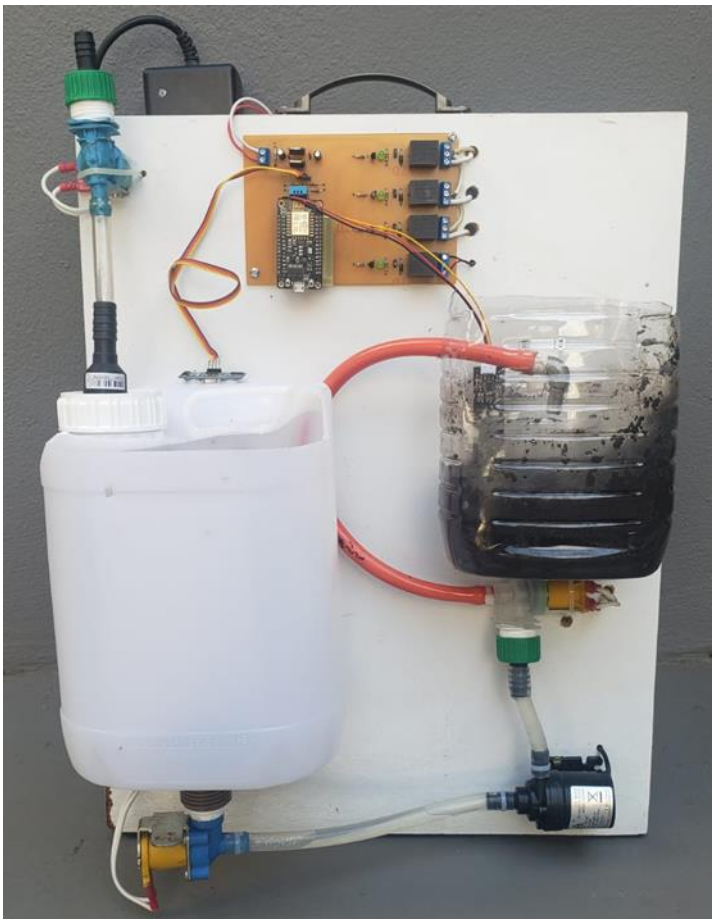
Figura 33: Integração MQTT Dash 2/2

Fonte: Autores

4. RESULTADOS E DISCUSSÃO

Ao final da montagem, foi possível obter um protótipo funcional que pôde validar o método de conexão ao protocolo MQTT para gestão de fornecimento de água e irrigação do solo. Obtendo com sucesso a integração do protocolo MQTT ao código fonte desenvolvido para automação do sistema de irrigação. Concluindo com considerações pontuais a respeito do uso do protocolo de comunicação como ferramenta facilitadora de implementação.

Figura 34: Protótipo Funcional



Fonte: Autores

4.1 Considerações

Como o protocolo de comunicação apresenta um funcionamento exclusivamente online, a possibilidade de queda de comunicação do dispositivo com a internet oferecia um risco de interferência no sistema de irrigação.

Durante os ensaios, foi possível observar que uma vez que o dispositivo perde conexão com o *broker*, o protocolo desencadeava uma série de erros no sistema, fazendo com que o código fonte travasse. Erro esse que tornaria inviável a implementação da solução. Porém, o mesmo pôde ser contornado com uma solução desenvolvida no próprio código, fazendo com que a falta de conexão não interfira no funcionamento do sistema de irrigação. Finalmente, foi possível obter um protótipo funcional, que integra o sistema de automação com a troca de informações entre dispositivos pelo *broker* MQTT.

5. CONCLUSÕES

Com a conclusão do projeto, foi possível observar com sucesso a integração do protocolo de comunicação MQTT, juntamente com o sistema automatizado de irrigação. Ainda a validação de forma experimental, utilizando uma representação do sistema de irrigação e seu reservatório. Sendo possível validar também as técnicas utilizadas para desenvolvimento dos circuitos impressos para desenvolvimento do protótipo e o funcionamento do método de simulação do sistema de irrigação. Apesar disso, deve-se considerar que os materiais utilizados no desenvolvimento do projeto deverão ser adaptados caso o interesse seja uma aplicação em escala real, ao invés de um protótipo.

Como ponto de destaque, é possível observar que as conexões digitais e analógicas presentes no microcontrolador NodeMCU limitam o escalonamento de áreas que podem ser gerenciadas para irrigação. Uma solução futura que poderá ser realizada é a integração de um multiplexador à entrada analógica, o que ampliaria a capacidade de sensores que poderiam ser utilizados. Sendo assim, seria possível manter o microcontrolador escolhido.

Outro ponto interessante, considerando um trabalho futuro, seria a utilização de um sensor ultrassônico com resistência à umidade. Uma opção ao HCSR04, que foi utilizado como forma de validação mínima, seria o JSN-SR04T, que possui resistência a líquidos, podendo inclusive ser utilizado em ambientes externos e sendo somente impróprio para uso submerso. O uso do componente elevaria significativamente o custo de implementação, porém se faz necessário devido à sua aplicação.

6. APÊNDICE

Nesse Apêndice é apresentado o código utilizado no projeto, bem como sua explicação. Inicialmente, são declaradas as bibliotecas e como cada pino é referido no programa:

```
/* Programa: Sistema Supervisório de Irrigação
   Automatizado e de Acesso Remoto
   Autor: Wellington Batista Fernandes e Artur Araujo Porto
   Descrição: Programa de automatização de irrigação através de
   Aplicativo em comunicação MQTT com o NodeMCU.
*********************************************************************/

//Lista de Bibliotecas
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Ultrasonic.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

//Lista de Definição de Pinos     PINOS  NOME
#define pino_bomba 15             //15    D8
#define pino_valvula_torneira 13  //13    D7
#define pino_valvula_tanque 12    //12    D6
#define DHTPIN 14                 //14    D5
//Sem uso                         //02    D4
//Sem uso                         //00    D3
#define pino_trigger 5            //05    D2
#define pino_echo 4               //04    D1
#define pino_valvula_planta1 16   //16    D0
#define pino_planta1 A0           //A0    A0
```

Logo após, temos a declaração de todas as variáveis utilizadas no nosso programa:

```
//Lista de Definição de Variaveis Globais
int distancia;                    //Variável do sensor Ultrassônico
int sensor_planta_1;              //Variável do sensor de Umidade
int alerta = 0;                   //Indica alerta do nivel do tanque
int irriga_planta_1 = 0;          //Indica se o sistema está irrigando
int planta1seca = 0;              //Indica que o solo está seco.
int modo = 0;                     //Modo de funcionamento  0=Auto 1=Manual
const int UMIDADE_MAXIMA = 712;   //Valor com solo muito seco
const int UMIDADE_MINIMA = 300;   //Valor com solo muito umido
const int NIVEL_MAXIMO = 24;      //Valor com tanque vazio
const int NIVEL_MINIMO = 7;       //Valor com tanque cheio
int leitura_planta1 = 0;          //Variável de mapeamento da Umidade
int nivel_tanque = 0;             //Variavel de mapeamento do Tanque
int manualplanta1 = 0;            //Variável para acionamento da irrigação manual
int manualtanque = 0;             //Variável para abastecimento manual
unsigned long previousMillis = 0;
const long interval = 1000;       //Variável para delay entre leituras do sistema
int ledState = LOW;               //Variável para delay entre leituras do sistema
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
#define DHTTYPE DHT11             //Definição do sensor de temperatura/umidade utilizado
DHT_Unified dht(DHTPIN, DHTTYPE);
uint32_t delayMS;
const char* ssid = "NOME DA REDE ";
const char* password = "SENHA DA REDE";
const char* mqtt_server = "ENDEREÇO DO BROKER MQTT";
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
int value = 0;
Ultrasonic ultrasonic(pino_trigger, pino_echo); //(5, 4)
```

Em seguida, temos o laço “setup_wifi” de configuração Wi-Fi. Nele, definimos o comportamento do NodeMCU durante uma tentativa de conexão Wi-Fi:

```
void setup_wifi() {
  delay(10);
  Serial.println("");
  Serial.print("Conectando com ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("IP: ");
  Serial.println(WiFi.localIP());
}
```

No laço “callback”, temos a configuração de cada modo de funcionamento do sistema e as mensagens a serem publicadas de forma correta em cada tópico.

```
void callback(char* topic, byte* payload, unsigned int length) {
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);}
  if ((char)payload[0] == 'A') {
    snprintf (msg, MSG_BUFFER_SIZE, "Automático");
    client.publish("casa/modo", msg);
    modo = 0;}
  if ((char)payload[0] == 'a') {
    snprintf (msg, MSG_BUFFER_SIZE, "Manual");
    client.publish("casa/modo", msg);
    client.publish("casa/Status", msg);
    modo = 1;}
  if ((char)payload[0] == 'O') {
    snprintf (msg, MSG_BUFFER_SIZE, "Irrigando Planta 1 Manualmente");
    client.publish("casa/Status", msg);
    manualplanta1 = 1;}
  if ((char)payload[0] == 'o') {
    snprintf (msg, MSG_BUFFER_SIZE, "Desligando Planta 1 Manualmente");
    client.publish("casa/Status", msg);
    manualplanta1 = 0;}
  if ((char)payload[0] == 'W') {
    snprintf (msg, MSG_BUFFER_SIZE, "Abrindo Válvula do Tanque Manualmente");
    client.publish("casa/Status", msg);
    manualtanque = 1;}
  if ((char)payload[0] == 'w') {
    snprintf (msg, MSG_BUFFER_SIZE, "Fechando Válvula do Tanque Manualmente");
    client.publish("casa/Status", msg);
    manualtanque = 0;}
}
```

No laço “reconnect”, temos a configuração da conexão com o Broker MQTT e a criação de um ID aleatório para o NodeMCU, assim como o comportamento em caso de queda de conexão.

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Aguardando conexão MQTT...");
    String clientId = "ESP8266Client";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("conectado");
      client.subscribe("casa/publisher");
    }
    else {
      Serial.print("falhou, rc=");
      Serial.print(client.state());
      Serial.println("tente novamente em 5s");
      delay(5000);
      break;
    }
  }
}
```

Podemos observar que em caso de falha, o leitor não deixa de executar suas funções, apenas aguardando 5 segundos para realizar uma nova tentativa de conexão.

No laço “leitura_sensor”, temos a leitura do sensor DHT11, que realiza as medições de temperatura e umidade no ar. Estas medições são publicadas nos tópicos determinados para visualização no dashboard.

```
void leitura_sensor() {
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  if (isnan(event.temperature)) {
    snprintf (msg, MSG_BUFFER_SIZE, "Erro na leitura da temperatura!");
    client.publish("casa/Status", msg);
  }
  else {
    sprintf(msg, "%f", event.temperature);
    dtostrf(event.temperature, 5, 2, msg);
    client.publish("casa/temperatura", msg);
  }
  dht.humidity().getEvent(&event);
  if (isnan(event.relative_humidity)) {
    snprintf (msg, MSG_BUFFER_SIZE, "Erro na leitura da umidade!");
    client.publish("casa/Status", msg);
  }
  else {
    sprintf(msg, "%f", event.relative_humidity);
    dtostrf(event.relative_humidity, 5, 2, msg);
    client.publish("casa/umidade", msg);
  }
}
```

Neste laço, temos a condição que em caso de erro na leitura de temperatura ou umidade do ambiente, ele envia uma mensagem de erro ao servidor indicando o erro.

No laço “nível_do_tanque”, temos o mapeamento da leitura do sensor HC-SR04, onde é transformada a leitura do sensor para uma escala de 0 até 100%.

```
void nivel_do_tanque() {
  distancia = ultrasonic.read();
  nivel_tanque = map(distancia, NIVEL_MINIMO, NIVEL_MAXIMO, 100, 0);
  if (nivel_tanque < 0) {
    nivel_tanque = 0;
  }
  if (nivel_tanque > 100) {
    nivel_tanque = 100;
  }
  sprintf(msg, "%f", nivel_tanque);
  dtostrf(nivel_tanque, 5, 2, msg);
  client.publish("casa/tanque", msg);
}
```

No laço “verifica_tanque”, temos os comportamentos do sistema de acordo com o nível de água no tanque.

```
void verifica_tanque() {
  if (nivel_tanque <= 10) {
    digitalWrite(pino_valvula_torneira, HIGH);
    snprintf (msg, MSG_BUFFER_SIZE, "T");
    client.publish("casa/torneira", msg);
    alerta = 2;
    irriga_planta_1 = 0;
    digitalWrite(pino_bomba, LOW);
    snprintf (msg, MSG_BUFFER_SIZE, "b");
    client.publish("casa/bomba", msg);
    digitalWrite(pino_valvula_tanque, LOW);
    digitalWrite(pino_valvula_planta1, LOW);
    snprintf (msg, MSG_BUFFER_SIZE, "i");
    client.publish("casa/irrigacao", msg);
    snprintf (msg, MSG_BUFFER_SIZE, "Nivel critico do tanque");
    client.publish("casa/Status", msg);
```

No caso do nível do tanque em 10% ou abaixo, por segurança, temos a desativação de todo o sistema de irrigação e a abertura do da válvula de abastecimento do tanque até que o nível esteja maior do que 10%.

```
    if (nivel_tanque >= 15) {
      alerta = 1;
      Serial.println("Nivel Satisfatório Atingido");
      snprintf (msg, MSG_BUFFER_SIZE, "O sistema foi reativado");
      client.publish("casa/Status", msg);
      irriga_planta_1 = 1;
    }
  }
```

Assim que nível do tanque chegar em 15% ou mais, temos a reativação de todo o sistema de irrigação e a abertura do da válvula de abastecimento do tanque até que o nível esteja maior do que 25%.

```
  else if (nivel_tanque <= 25) {
    Serial.println("");
    Serial.println("Reservatorio Baixo");
    digitalWrite(pino_valvula_torneira, HIGH);
    snprintf (msg, MSG_BUFFER_SIZE, "A valvula da Torneira foi aberta");
    client.publish("casa/Status", msg);
    snprintf (msg, MSG_BUFFER_SIZE, "T");
    client.publish("casa/torneira", msg);
    alerta = 1;
  }
  else {
    digitalWrite(pino_valvula_torneira, LOW);
    snprintf (msg, MSG_BUFFER_SIZE, "Reservatório OK");
    client.publish("casa/Status", msg);
    snprintf (msg, MSG_BUFFER_SIZE, "t");
    client.publish("casa/torneira", msg);
    alerta = 0;
  }
}
```

Em caso de nível de água menor do que 25%, não é necessário paralisação do sistema de irrigação, apenas é realizado a abertura da válvula de abastecimento até que o nível do tanque supere 25%. Caso o sistema não detecte que o nível do tanque está abaixo de 25%, apenas é enviado uma mensagem indicando que o nível do tanque está normal.

No laço “planta1”, temos a leitura do sensor capacitivo de umidade do solo, assim como o mapeamento da escala de 0 até 100%.

```
void planta1() {
  sensor_planta_1 = analogRead(pino_planta1);
  leitura_planta1= map(sensor_planta_1, UMIDADE_MINIMA, UMIDADE_MAXIMA, 100, 0);
  if (leitura_planta1 < 0) {
    leitura_planta1 = 0;
  }
  if (leitura_planta1 > 100) {
    leitura_planta1 = 100;
  }
  sprintf(msg, "%f", leitura_planta1);
  dtostrf(leitura_planta1, 5, 2, msg);
  client.publish("casa/solo", msg);
}
```

Após a leitura do solo, temos o código com o comportamento do sistema de acordo com a leitura do sensor. Caso o sistema detecte que a umidade do solo está abaixo de 50%, será iniciada a irrigação até a umidade do solo ultrapasse 50%.

```
void irrigacao1() {
  if (alerta == 2) {
    digitalWrite(pino_valvula_tanque, LOW);
    digitalWrite(pino_bomba, LOW);
    digitalWrite(pino_valvula_planta1, LOW);
  }
  else {
    if (leitura_planta1 >= 0 && leitura_planta1 <= 50) {
      irriga_planta_1 = 1;
      digitalWrite(pino_valvula_tanque, HIGH);
      digitalWrite(pino_bomba, HIGH);
      digitalWrite(pino_valvula_planta1, HIGH);
      snprintf (msg, MSG_BUFFER_SIZE, "I");
      client.publish("casa/irrigacao", msg);
      snprintf (msg, MSG_BUFFER_SIZE, "Irrigando Planta 1");
      client.publish("casa/Status", msg);
      snprintf (msg, MSG_BUFFER_SIZE, "B");
      client.publish("casa/bomba", msg);
    }
```

```
    if (leitura_planta1 >= 51 && irriga_planta_1 == 1) {
      irriga_planta_1 = 0;
      digitalWrite(pino_bomba, LOW);
      digitalWrite(pino_valvula_tanque, LOW);
      digitalWrite(pino_valvula_planta1, LOW);
      snprintf (msg, MSG_BUFFER_SIZE, "i");
      client.publish("casa/irrigacao", msg);
      snprintf (msg, MSG_BUFFER_SIZE, "Irrigação Finalizada");
      client.publish("casa/Status", msg);
      snprintf (msg, MSG_BUFFER_SIZE, "b");
      client.publish("casa/bomba", msg);
      planta1seca = 0;
    }
    delay(100);
  }
}
```

Assim que a umidade ultrapassar 51% de umidade, a irrigação será encerrada.

No laço “setup”, temos as configurações dos pinos de entrada e saída, comunicação serial, iniciação do sensor DHT11, iniciação do Wi-Fi e do protocolo MQTT.

```
void setup() {
  pinMode(pino_bomba, OUTPUT);
  pinMode(pino_valvula_torneira, OUTPUT);
  pinMode(pino_valvula_tanque, OUTPUT);
  pinMode(pino_valvula_planta1, OUTPUT);
  pinMode(pino_planta1, INPUT);
  Serial.begin(9600);
  dht.begin();
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  dht.humidity().getSensor(&sensor);
  delayMS = sensor.min_delay / 1000;
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
```

Após todas as configurações dos sensores, conexão Wi-Fi e do protocolo MQTT, podemos entrar no laço loop, o laço principal do programa.

No laço loop, temos a variável modo que define o funcionamento do sistema entre automático e manual. Em caso automático, é realizado a chamada de todas as funções automaticamente, realizando a leitura dos sensores e todo o sistema de irrigação de forma automática.

```
void loop() {
  delay(100);
  unsigned long currentMillis = millis();
  if (modo == 0) {
    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;     //
      if (ledState == LOW) {
        ledState = HIGH;
        leitura_sensor();
        nivel_do_tanque();
        verifica_tanque();
        planta1();
        irrigacao1();
      }
      else {
        ledState = LOW;
      }
      delay(200);
    }
  }
```

Também é possível acionar o modo manual do sistema, de modo que o acionamento do sistema seja realizado via qualquer dispositivo ligado ao broker MQTT, facilitando uma possível manutenção, calibração ou ajustes do sistema hidráulico.

```
  if (modo == 1) {
    leitura_sensor();
    planta1();
    nivel_do_tanque();
    if (manualplanta1 == 1) {
      digitalWrite(pino_valvula_tanque, HIGH);
      digitalWrite(pino_bomba, HIGH);
      digitalWrite(pino_valvula_planta1, HIGH);
    }
    else {
      digitalWrite(pino_valvula_tanque, LOW);
      digitalWrite(pino_bomba, LOW);
      digitalWrite(pino_valvula_planta1, LOW);
    }
    if (manualtanque == 1) {
      digitalWrite(pino_valvula_torneira, HIGH);
    }
    else {
      digitalWrite(  pino_valvula_torneira , LOW);
    }
  }
```

Por fim, temos uma condição que checa se o sistema está conectado ao broker MQTT, realizando uma tentativa de reconexão em caso de interrupção da conexão.

```
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}
```

REFERÊNCIAS

AMERICANAS, Bomba de Água. 2022. Fotografia. Disponível em: https://www.americanas.com.br/produto/1582559840?pfm_carac=motor-de-bomba-submersa

BYTEFLOP, ESP8266. 2022. Fotografia. Disponível em: https://www.byteflop.com.br/modulo-wifi-nodemcu-esp8266-ch340. Acesso em: 22 fev. 2022.

CURTOCIRCUITO, Sensor Capacitivo de umidade. 2022. Fotografia. Disponível em: https://www.curtocircuito.com.br/sensor-de-umidade-do-solo-capacitivo.html. Acesso em: 10 maio 2022.

FILIPEFLOP, Diagrama de tempo HC-SR04. 23 jul. 2011. Fotografia. Disponível em: https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino/#:~:text=O%20funcionamento%20do%20HC%2DSR04,sensor%20e%20o%20objeto%20detectado. Acesso em: 16 mar. 2022.

FILIPEFLOP, Sensor DHT11. 5 ago. 2013. Fotografia. Disponível em: https://www.filipeflop.com/blog/monitorando-temperatura-e-umidade-com-o-sensor-dht11/. Acesso em: 18 abr. 2022.

FILIPEFLOP, Sensor HC-SR04. 2022. Fotografia. Disponível em: https://www.filipeflop.com/produto/sensor-de-distancia-ultrassonico-hc-sr04/. Acesso em: 24 mar. 2022.

GITLAB UFRGS, ESP8266 Pinout. 20 maio 2022. Fotografia. Disponível em: https://git.cta.if.ufrgs.br/CosmicPampa/CosmicPampa-META/-/issues/3. Acesso em: 15 mar. 2022.

MERCADOLIVRE, Válvula Solenóide. 2022. Fotografia. Disponível em: https://produto.mercadolivre.com.br/MLB-2610377462-valvula-solenoide-maquina-lavar-brastemp-110v-_JM

MUNDO PROJETADO, Funcionamento do sensor a partir do solo. 2022. Fotografia. Disponível em: https://mundoprojetado.com.br/sensor-de-umidade-capacitivo-para-solo/. Acesso em: 18 maio 2022.

JANK, Marcos Sawaya; NASSAR, André Meloni; TACHINARDI, Maria Helena. Agronegócio e comércio exterior brasileiro. Revista USP, n. 64, p. 14-27, 2005.

LEI Nº 11.326, Política Nacional da Agricultura Familiar e Empreendimentos Familiares Rurais. Brasil, 24 jul. 2006. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2006/lei/l11326.htm. Acesso em: 13 mar. 2022.

EMBRAPA, Módulos Fiscais. Brasil, 2012. Disponível em: https://www.embrapa.br/codigo-florestal/area-de-reserva-legal-arl/modulo-fiscal. Acesso em: 13 mar. 2022.

NAKATANI, Alessandro Massayuki; GUIMARÃES, Anderson Valenga; NETO, Vicente Machado. Medição com sensor ultrassônico HC-SR04. In: International Congress on Mechanical Metrology. Gramado. 2014.

OASIS, Open. Brasil, 2022. Disponível em: https://www.oasis-open.org/. Acesso em: 3 mar. 2022.

OLIVEIRA, Ricardo Rodrigues. Uso do microcontrolador ESP8266 para automação residencial. Rio de Janeiro: UFRJ Escola Politécnica, 2017.

OSHWA, Open Source Hardware Association. Brasil, 16 mar. 2022. Disponível em: https://www.oshwa.org/. Acesso em: 5 abr. 2022.

TORRES, Andrei BB; ROCHA, Atslands R.; DE SOUZA, José Neuman. Análise de desempenho de *brokers* mqtt em sistema de baixo custo. In: Anais do XV Workshop em Desempenho de Sistemas Computacionais e de Comunicação. SBC, 2016. p. 2804-2815