CENTRO UNIVERSITÁRIO DO SUL DE MINAS
BACHARELADO EM ENGENHARIA ELÉTRICA
WILLIAM CRUZ DE OLIVEIRA

CONTROLE DE IRRIGAÇÃO BASEADO NA UMIDADE DO SOLO UTILIZANDO SISTEMA EMBARCADO

Varginha
2018

WILLIAM CRUZ DE OLIVEIRA

CONTROLE DE IRRIGAÇÃO BASEADO NA UMIDADE DO SOLO UTILIZANDO SISTEMA EMBARCADO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica do Centro Universitário do Sul de Minas como pré-requisito para obtenção de grau de bacharel sob a orientação do Prof. Esp. Eduardo Henrique Ferroni e coorientação do Prof. Esp. Matheus Henrique Pereira.

Varginha

2018

WILLIAM CRUZ DE OLIVEIRA

CONTROLE DE IRRIGAÇÃO BASEADO NA UMIDADE DO SOLO UTILIZANDO SISTEMA EMBARCADO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica do Centro Universitário do Sul de Minas como pré-requisito para obtenção do grau de bacharel pela banca examinadora composta pelos membros:

Aprovado em / /

Prof. Esp. Eduardo Henrique Ferroni

Prof. Esp. Marcelo Pereira Gonçalves

Prof. Esp. Paulo Roberto de Paiva Novo

OBS.:

Dedico este trabalho à minha esposa Elaine e ao meu filho João Lucas, que estiveram do meu lado e me apoiaram, no momento mais difícil da minha vida, mostrando que era possível acreditar e ter esperança em Deus para dias melhores.

AGRADECIMENTOS

Agradeço primeiro à Deus pelas oportunidades da vida e por sua presença em todos os momentos, difíceis e alegres que me fizeram crescer e acreditar mais na sua grandeza e bondade, fortalecendo a minha saúde e disposição para superar todas as etapas.

À minha esposa, Elaine, que sempre está ao meu lado, em todos os momentos, me apoiando, incentivando e motivando, acreditando na minha capacidade. Obrigado pela compreensão, companheirismo e amor de esposa. Você é uma grande mulher e o pilar da minha vida.

Ao meu lindo filho, João Lucas, que veio como uma benção de Deus na minha vida, me realizando como pai. A ele minhas desculpas,

pelos momentos ausentes. Te amo de forma singular.

Aos meus pais, João da Cruz (em memória) e Maria do Carmo, pela vida, incentivo e educação.

À EPTV pelo incentivo e pela oportunidade de crescimento profissional.

Ao Clóvis, gestor do setor de Engenharia da EPTV, que me depositou sua confiança e me apoiou.

Aos meus colegas de trabalho, da EPTV, pelo apoio.

Ao professor e meu orientador neste trabalho, Eduardo Ferroni, pelas dicas e sugestões que foram de total importância para que este trabalho se concretizasse.

Ao professor e coorientador Matheus Henrique, que além de ser meu irmão de coração, se demonstrou um grande companheiro e incentivador.

A professora Luciene pela atenção e dicas valiosas na formatação deste trabalho.

Às demais pessoas, que de uma ou de outra maneira, apesar de não serem citadas aqui, deram créditos a este trabalho. A minha eterna gratidão a todos!

“A percepção do desconhecido é a mais fascinante das experiências. O homem que não tem os olhos abertos para o misterioso passará pela vida sem ver nada.”

Albert Einstein

RESUMO

Este trabalho tem como objetivo fazer o uso dos recursos tecnológicos para aprimorar o trabalho de produtores agrícola que lidam com a terra, seja no campo ou na cidade, com a capacidade de monitorar a umidade do solo e atuar, de forma sistemática em um sistema de irrigação de maneira automatizada. Ele faz parte de um novo conceito onde coisas e objetos possam estar conectados à *Internet* possibilitando infinitas possibilidades, denominado *Internet of Things* (IoT). Desta forma aplicar a tecnologia para trazer benefícios aos agricultores, visando economia no uso dos recursos hídricos e rentabilidade em produtos de melhor qualidade. A irrigação é a técnica utilizada em centenas de anos por agricultores para umedecer a terra, para se cultivar determinado produto em regiões de poucas chuvas ou em época de escassez. Se este método fosse realizado conscientemente o suficiente não traria tanto impacto ambiental, o que muito se discute em debates e seminários promovidos por várias categorias de instituições, governamentais ou não, e que na prática muito pouco é realizado e não garante resultados satisfatórios. A utilização desordenada dos recursos hídricos provoca impactos econômicos e sociais, mas estes problemas podem ser sanados através de sistemas automatizados. Com o avanço em pesquisa e tecnológica, a população está cada vez mais interligada a grande rede que é a *Internet*, esteja ela no campo ou nos grandes centros, elas se dispõe de inúmeras possibilidades para se manter conectada, o que facilita a elaboração de projetos com o intuito de facilitar as tarefas cotidianas. A intenção deste trabalho é fazer o uso destes recursos para disponibilizar ao agricultor uma ferramenta que irá auxilia-lo no cultivo de sua produção, o que agregam vários outros benefícios. O uso da *Internet* e sistemas embarcados equipados com sensores e atuadores podem trazer resultados satisfatórios para este produtor, mas principalmente visando uma contribuição ao meio ambiente.

Palavras-chave: Tecnologia. *Internet* das Coisas. Produtores Agrícola. Irrigação. Meio Ambiente.

ABSTRACT

This work has as objective to make use of technological resources to improve the work of the agricultural producers that labor with the ground, whether in the countryside or in the city, with the ability to monitor soil moisture and to act systematically in a irrigation in an automated way. It is part of a new concept where things and objects can be connected to the Internet allowing infinite possibilities, called Internet of Things (IoT). In this way to apply the technology to bring benefits to the agriculturist, aiming at saving in the use of water resources and profitability in products of better quality. The Irrigation is the technique used by agriculturist to moisten the ground for hundreds of years to cultivate a particular product in regions of the few rains or in times of scarcity. If this method were carried out consciously enough, it would not have as much environmental impact, and that much discussed in debates and seminars promoted by various categories of institutions, governmental or otherwise, and that in practice very little is done and does not guarantee satisfactory results. The disorganized use of water resources causes economic and social impacts, but these problems can be resolved through automated systems. With the advancement in research and technology, the population is increasingly connected to the great network that is the Internet, whether it is in the field or in large centers, they have many possibilities to stay connected, which facilitates the elaboration of projects with the purpose of facilitating the daily tasks. The intention of this work is to make use of these resources to make available to the agriculturist a tool that will help him in the cultivation of his production, which adds several other benefits. The use of the Internet and embedded systems equipped with sensors and actuators can bring satisfactory results to this producer, but mainly aiming a contribution to the environment.

Keywords: *Technology. Internet of Things. Agricultural Producers. Irrigation. Environment.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A	Ampères
AD	Analógico / Digital
ADC	Conversor Analógico Digital
ANA	Agência Nacional das Águas
ANSI	*American National Standard Institute*
AP	*Access Point*
ARM	*Advanced RISC Machine*
ASP	*Active Server Pages*
BBC	*British Broadcasting Corporation*
BD	Banco de Dados
CSS	*Cascading Style Sheets*
CVA	Conteúdo Volumétrico
CONAMA	Conselho Nacional do Meio Ambiente
DCL	*Data Control Language*
DDL	*Data Definition Language*
DML	*Data Manipulation Language*
DOM	*Document Object Model*
EEPROM	*Eletrical Erasable Programmable Read Only Memory*
EPROM	*Erasable Programmable Read Only Memory*
FAO	Organização das Nações Unidas para a Alimentação e Agricultura
GND	*Ground*
GPS	*Global Positioning System*
HTML	*HyperText Markup Language*
HTML5	*HyperText Markup Language* versão 5
HTTP	*Hypertext Transfer Protocol*
ID	Identificador
I2C	*Inter-Integrated Circuit*
ICSP	*In Circuit Serial Programming*
IDE	*Integrated Development Environment*
IEC	*International Electrotechnical Commission*
IoT	*Internet of Things*
IP	*Internet Protocol*

IP	*Ingress Protect*
ISM	*Industrial Scientific and Medical*
ISP	*Internet Service Provider*
JSP	*Java Server Pages*
LCD	*Liquid Crystal Display*
LED	*Light Emitting Diode*
MIT	*Massachusetts Institute of Technology*
NC	*Normally Close*
NO	*Normally Open*
ONU	Organização das Nações Unidas
OS	*Operational System*
Pa	*Pascal*
PC	*Personal Computer*
PHP	*Hypertext Preprocessor*
PROM	*Programmable Read Only Memory*
PWM	*Pulse Width Modulation*
RDBMS	Sistema de Gerenciamento de Banco de Dados Relacional
RF	Rádio Frequência
RISC	*Reduced Instruction Set Computer*
RSSF	Redes de Sensores sem Fio
RTC	*Real Time Clock*
SCL	*Serial Clock*
SDA	*Serial Data*
SGBD	Sistema de Gerenciamento de Banco de Dados
SPI	*Serial Peripheral Interface*
SRAM	*Static Random Access Memory*
SQL	*Structured Query Language*
STA	*Station*
TCP	*Transmission Control Protocol*
TFT	*Thin Film Transistor*
TI	Tecnologia da Informação
TIP	Transistor de Potência
TTL	*Transistor Transistor Logic*
UART	*Universal Asynchronous Receiver Transmitter*

UDP	*User Datagram Protocol*
USB	*Universal Serial Bus*
VAC	Voltagem em Corrente Alternada
VBE	*Base-Emissor Voltage*.
VCC	Voltagem em Corrente Contínua
VDC	*Voltage Direct Current*
W3C	*World Wide Web Consortium*
WAP	*WiFi Protected Access*
WEP	*Wired Equivalent Privacy*
WIFI	*Wireless Fidelity*
WWW	*World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

Com o avanço tecnológico em vários setores torna-se possível expandir o ângulo de visão no que se diz respeito a criar soluções para problemas comuns em diversas áreas.

Na agricultura, o ponto de maior relevância é a irrigação, que necessita de adequada disponibilidade e de uma água de boa qualidade. Estudos realizados por órgãos especializados buscam uma maior eficiência para esta técnica como a Agência Nacional das Águas (ANA) (AGÊNCIA NACIONAL DAS ÁGUAS, 2017).

A preocupação com a irrigação é com o excesso de água que é utilizada, não sendo esta aproveitada na sua totalidade. Esta sobra retornar aos corpos d´água, superficiais ou subterrâneos com insumos, sais solúveis e defensivos agrícolas, gerando um alto nível de desperdício e contaminando o meio ambiente (ANA, 2017).

Um dos incentivo ao desenvolvimento deste trabalho é pelo fato que a eficiência do uso da água passa por exigências e estímulos legais o que leva a elaboração de projetos que incorporem equipamentos e métodos de irrigação mais eficientes que sua vez tem prioridade no seu licenciamento (Resolução CONAMA nº 284/2001); (ANA, 2017).

Dentre as várias opções estudadas para reduzir o consumo de água na irrigação, é considerada a utilização de um sistema independente que possa monitorar a umidade do solo e mantê-la dentro do nível adequado para cada tipo de plantação, fazendo o uso somente da quantidade necessária de água no plantio visando também na qualidade da produção.

O mundo que conhecemos hoje já sofreu várias transformações e muitas delas veio a acrescentar na produtividade dos seres humanos, facilitando os trabalhos do nosso cotidiano, agilizando as produções, organizando as tarefas e agregando resultados satisfatórios em todos os seguimento. Com um planeta conectado, um conceito muito importante surgiu, denominado de IoT, uma iniciativa do *Massachusetts Institute of Tecnology* (MIT) nos Estados Unidos, possibilitando que coisas, objetos, sensores, geladeiras, carros, enfim, qualquer dispositivo que possibilite coletar, analisar e distribuir dados que possam ser transformados em informação e com isso capazes de serem controlados remotamente, envolvido em uma revolução indústria que já está na sua quarta geração.

Para que isso se torne possível, com uma solução acessível e funcional, se propõe o uso de interfaces computacionais que possa auxiliar tanto o controle, quanto na monitoração dos pontos de irrigação, o microcontrolador Arduino, que associado a uma extensa gama de módulos e sensores devidamente programados, possam trabalhar a favor de tal objetivo, estando

conectado, fornecendo e trabalhando com dados necessários para mais um dispositivo conectado.

Outros meios computacionais e tecnológicos podem se unir a esta ferramenta para implementar o seu uso como, a centralização das informações e configuração em um servidor, uma comunicação no padrão de rede sem fio (WIFI) e um sistema de sensoriamento alimentado por placas fotovoltaicas.

Com este propósito pretende-se atingir o objetivo de forma satisfatória para atender e sanar o problema de forma sustentável.

2 RECURSO HÍDRICO NO BRASIL

Com o avanço tecnológico em vários setores se torna possível expandir o ângulo de visão no que se diz respeito a dar soluções para problemas comuns em diversas áreas. Um problema observado é o alto consumo de água doce no setor agrícola de uma forma geral. Segundo a Organização das Nações Unidas para a Alimentação e Agricultura (FAO, 2015), é a atividade agropecuária a principal responsável pelo uso da água. A agricultura é o setor da economia que mais necessita da imposição de medidas de redução do consumo de água, pois cerca de 60% de toda a água empregada na irrigação estaria sendo desperdiçada. Assim, os mesmos estudos apontam que uma redução de 10% dessa perda seria o suficiente para abastecer o dobro da população mundial atual, em termos de média estatística De acordo com a entidade, 70% de toda a água consumida no mundo é utilizada na irrigação das lavouras, número que se eleva para 75% no caso do Brasil, sendo um país com forte produção nesse setor da economia (ANA, 2017).

Para a sobrevivência do homem e de todos os seres vivos neste planeta, temos a água como um recurso natural indispensável. É o líquido fundamental existente na natureza e importante para a correta absorção de nutrientes do solo pelas plantas, além de imprescindível às formações hídricas atmosféricas, influenciando o clima das regiões. No ser humano, é responsável por aproximadamente três quartos de sua constituição. Este rico recurso da natureza encontra-se cada vez mais insuficiente e escasso pelas ações descontroladas do homem nas bacias hidrográficas, comprometendo a sua qualidade (FERREIRA, 2011).

A carência de água pode ser um dos principais fatores que impactam diretamente ao desenvolvimento de alguns países, pois o modelo tecnológico até então elaborado com base na exploração indiscriminada dos recursos naturais, está esgotado, segundo Ferreira (2011).

A água tem sido considerada, um recurso escasso e estratégico, por questão de segurança nacional e por seus valores social, econômico e ecológico (ANA, 2016).

A ideia que a grande maioria das pessoas faz com relação à água, é a de que ela é infinitamente abundante e sua renovação natural; no entanto, ocupando 71% da superfície do planeta, 97% deste total se constituem águas salgadas, 2,07% são águas doces em geleiras e calotas polares (água em estado sólido) e apenas 0,63% restam de água doce não totalmente aproveitados por questões de inviabilidade técnica, econômica e financeira e de sustentabilidade ambiental (ANA, 2016).

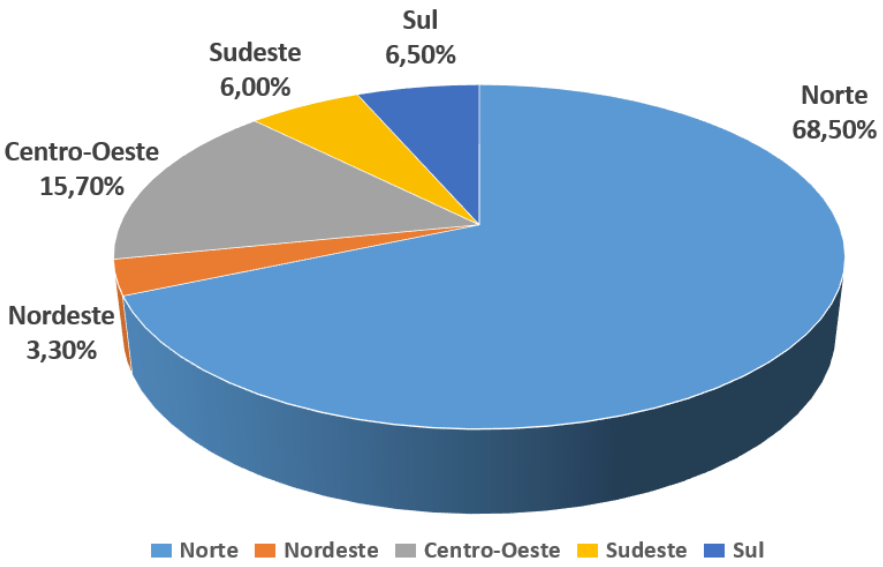
O continente da América Latina conta com abundantes recursos hídricos, porém existem consideráveis diferenças entre as distintas regiões nas quais os problemas de água se devem,

sobretudo, ao baixo rendimento de utilização, gerenciamento, contaminação e degradação ambiental. A Argentina, o Peru e o Chile já enfrentam sérios problemas de disponibilidade e contaminação da água por efluentes agroindustriais que são descarregados em canais de irrigação. A situação brasileira não é de tranquilidade, embora seja considerado um país privilegiado em recursos hídricos, enquanto conflitos de quantidade e déficit de oferta já são realidade. Outra questão se refere ao desperdício de água, estimado em 40%, por uso predatório e irracional, enquanto a escassez é cada vez mais grave na região Nordeste, onde a sobrevivência, a permanência da população e o desenvolvimento agrícola dependem essencialmente da oferta de água, afirma FAO (2015).

2.1 Irrigação no Brasil e o desperdício

O último levantamento revela que algo em torno de 75% da água captada no Brasil vai para a produção agrícola, segundo ANA (2016), mas esse consumo envolve diversas variáveis e, segundo especialistas consultados pela subsidiária da *British Broadcasting Corporation* (BBC) no Brasil, que atua como provedor mundial de notícias informa que ainda há desperdício significativo no setor e muito que fazer para economizar água. Os analistas concordam em uma coisa: o Brasil tem água o bastante para todos, mas precisa aprender a geri-la de forma mais eficiente e combater os desperdícios.

Figura 01. Distribuição dos recursos hídricos no Brasil

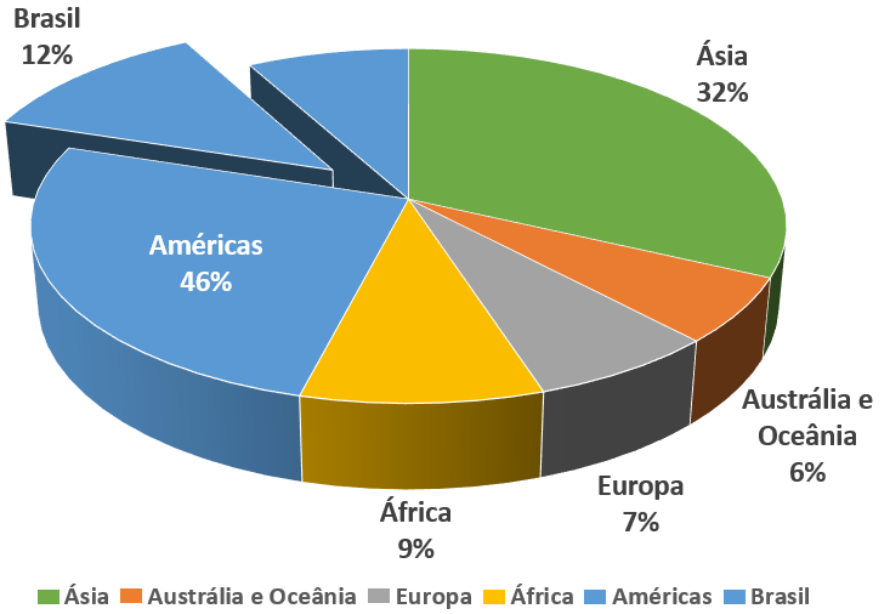


Fonte: (AGÊNCIA NACIONAL DAS ÁGUAS, 2016).

A Figura 01, indica que a região Norte é a que mais possui recursos hídricos e, no entanto, é a menos habitada, com apenas 7% da população brasileira. A região Nordeste, por sua vez, é a que possui menos água, concentrando-se mais nas áreas litorâneas da Zona da Mata e também no Meio Norte. Já as regiões Sul e Sudeste contam, igualmente, com um montante bastante limitado ao passo em que seus níveis de consumo são bastante acentuados, segundo a ANA (2016).

O Brasil tem posição privilegiada no mundo, em relação à disponibilidade de recursos hídricos, o que representa aproximadamente a 12% da disponibilidade mundial. A Figura 02 mostra a distribuição deste recurso no planeta.

Figura 02 - Distribuição de água doce no mundo



Fonte: (AGÊNCIA NACIONAL DAS ÁGUAS, 2016).

Por isso, o volume distribuído por pessoa é 19 vezes superior ao mínimo estabelecido pela Organização das Nações Unidas (ONU). No entanto, esse recurso vital não chega para todos os brasileiros na mesma quantidade e regularidade.

Não existe um método que seja melhor ou mais eficiente para a irrigação que estabeleça um compromisso com o meio ambiente. O foco dos produtores agrícolas é a produção, o produto final, independente de como seja a sua aplicação, causando impacto ambiental ou não, reforça Ferreira (2011).

Estudando cada método, pode-se definir soluções baseadas em tecnologia eficientes para o controle e monitoração dos processos de produção e evitar ou minimizar o desperdício.

2.2 Métodos de irrigação

Irrigação é uma técnica artificial de aplicação de água sobre o solo para fins de produção agrícola, para alcançar maiores rendimentos e atender às demandas do mercado, sazonal, especialmente em área onde haja escassez de chuva ou produção controlada de alimentos. A decisão de qual sistema de irrigação é melhor para o cultivo requer conhecimento de equipamentos, projeto de sistemas executado por especialistas, tipo de plantação e composição do solo. No Brasil, existem diversos sistemas para a irrigação, mas controla-los para que sejam utilizados de forma correta, ajuda no desenvolvimento sustentável e evita o que hoje é a maior preocupação que é o desperdício de água e degradação do meio ambiente. Os principais métodos de irrigação são por aspersão, por micro aspersão, localizada e por gotejamento.

2.2.1 Irrigação por aspersão

Irrigação por aspersão é um método de aplicação de água ao solo semelhante à precipitação. Esta água é distribuída através de um sistema de tubos, geralmente por bombeamento, e então é pulverizado no ar, na forma de minúsculas gotículas de água, irrigando toda a superfície do solo através das cabeças de pulverização, Figura 03, de modo que estas gotas caiam sobre o solo e a plantação de forma uniforme e constante por um determinado tempo, além da possibilidade de se adicionar fertilizantes e pesticidas. Os Aspersores, como são chamados, fornecem cobertura eficiente para área de pequeno e médio tamanho e são adequados para uso em todos os tipos de propriedades. Também é adaptável a quase todos os solos irrigáveis, uma vez que os Aspersores se adaptam facilmente em qualquer terreno disponíveis em uma ampla faixa de capacidade de volumes de água (TESTEZLAF, 2011).

Figura 03 – Irrigação por aspersão

Fonte: (HASS, 2017).

Os Aspersores, de acordo com a Figura 04, podem ser usados em uma configuração de posicionamento fixo, em que instala-se a quantidade necessário de cabeças de pulverização para atender uma certa região ou podem ser usados em uma configuração de movimentação de conjuntos, em que as linhas laterais são operadas e movidas em intervalos de tempo pré determinado dependendo da necessidade. Os sistemas de conjuntos sólidos custam mais para instalar, mas têm requisitos de mão-de-obra mais baixos e podem ser automatizados. O custo de equipamento e instalação por hectare de sistemas de movimento de conjuntos é menos dispendioso, mas sua operação exige mais mão-de-obra, pois eles não podem ser totalmente automatizados.

Figura 04 – Cabeça de pulverização

Fonte: Adaptado de (Makino, 2015).

2.2.2 Irrigação por micro aspersão

A irrigação por micro aspersão, conforme Figura 05, destinam-se a fornecer irrigação usando gotículas de água muito finas. A instalação é feita com defletores rotativos, conforme Figura 06, também conhecido como rotor ou dançarina, que ajuda a fornecer uma maior cobertura de diâmetro, menor taxa de precipitação do que os difusores, maior tamanho de gotas e melhor distribuição de água, principalmente de forma uniforme na distribuição.

Figura 05 – Irrigação por micro aspersão

Fonte: (Makino, 2015).

A principal diferença com a nebulização é que o micro aspersor projeta água em minúsculos jatos de água para a planta, em vez de fornecê-la de maneira nebulizada, e, por sua vez, possui elementos rotativos que distribuem água para a toda a superfície.

Figura 06 – Micro aspersor ou bailarina

Fonte: Adaptado de (Makino, 2015).

2.2.3 Irrigação localizada

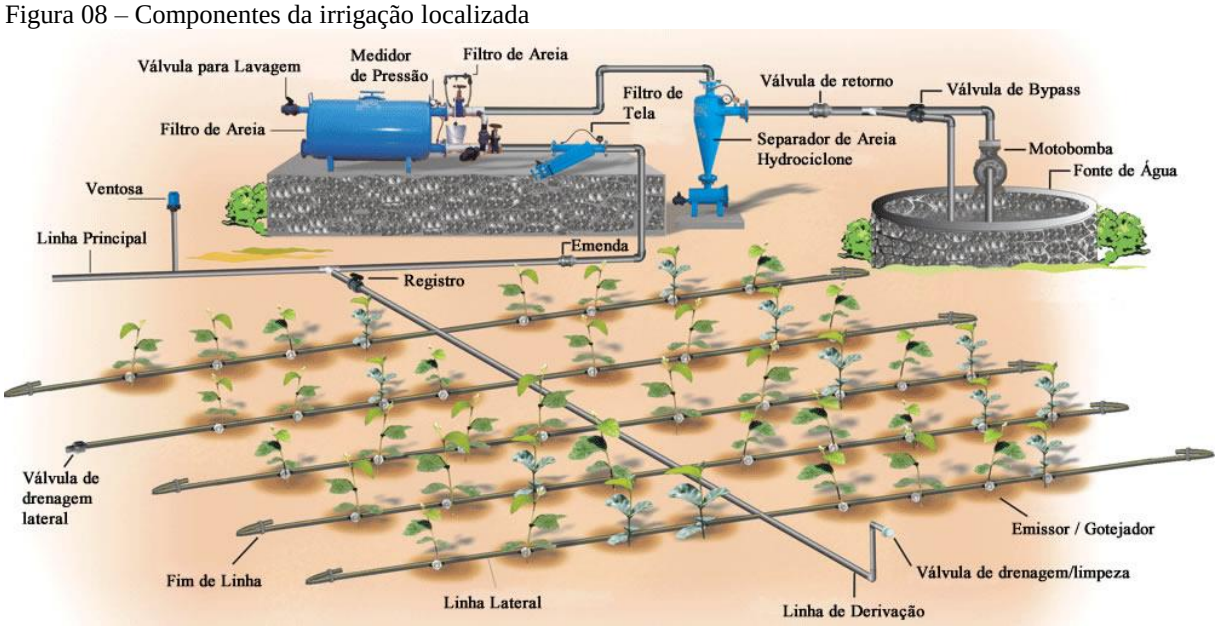
A irrigação localizada é um método de aplicação de água que resulta em irrigar apenas uma pequena área da superfície do solo, perto da base da planta concentrando a umidade somente na região da raiz. O fluxo em que se aplica a água geralmente é baixo, em pequenas quantidades e de forma intermitente, que pode ser acima ou abaixo da superfície do solo. Os dispositivos utilizados para fazer esta aplicação são tubos perfurados, orifícios ou bocais, de acordo com a Figura 07.

Figura 07 – Irrigação localizada

Fonte: (Oliveira, 2018).

Os principais componentes de um sistema de irrigação localizada são, o suprimento de água, reguladores de vazão e pressão, sistema de filtragem, as linhas principais e secundárias, distribuidores. A Figura 08 mostra alguns componentes do um sistema da irrigação localizada.

Figura 08 – Componentes da irrigação localizada



Fonte: Adaptado de (Oliveira, 2018).

As principais vantagens dos sistemas de irrigação, segundo Testezlaf (2017), são:

a) Faz um melhor aproveitamento dos recursos hídricos, pois irriga apenas a área ao redor da planta;
b) Proporciona um aumento na produção, melhorando a qualidade do produto, devido ao fato da umidade permanecer razoavelmente constante (próxima à capacidade de campo) e da distribuição ao longo da linha de cultivo ser mais uniforme.

c) Evita uma maior acidência de salinidade na plantação, pois devido à maior frequência na aplicação da água, maior será o período de umidade do solo, os sais são mantidos em maior diluição na água do solo.
d) Possibilita a aplicação de produtos químicos (fertilizantes, inseticidas, fungicidas) que podem ser diluídos na água, o que acarreta em redução na mão-de-obra, e na quantidade de insumos utilizados devido ao aumento da eficiência de aplicação desses produtos.
e) Utiliza pouca energia elétrica devido operar em baixas pressões, pouca vazão e curtos períodos de operação.

As desvantagens deste sistema é que ele possui limitações que inviabiliza que agricultores o utilizam, afirma Testezlaf (2017), que são:

a) Tem um investimento inicial elevado comparado a outros sistemas.
b) Causa muito entupimento, devido ao pequeno diâmetro dos emissores, causado principalmente por partículas de areia, fertilizantes, algas, bactérias, óxido de ferro e precipitados químicos, tornando-se necessário a manutenção periódica e o tratamento da água de irrigação.
c) Se projetado inadequadamente, a uniformidade de distribuição dos emissores pode ser afetada em áreas excessivamente declivosas, onde os emissores podem apresentar variações de vazão acima do recomendado em norma de projeto.
d) As operações de capina nas linhas de cultivo podem ser dificultadas pela presença das tubulações na superfície do solo.

O sistema de irrigação localizada por gotejamento compreende os sistemas onde a aplicação da água e de produtos químicos é realizada na forma de gotas por uma fonte pontual, denominado gotejador, conforme a Figura 09. Esses emissores operam com pressões que variam entre 50 a 200 kPa e vazões na ordem de 0,5 a 12 L h^{-1} (TESTEZLAF, 2017).

Figura 09 – Irrigação localizada por gotejamento

Fonte: (Oliveira, 2018).

2.2.4 Automatizando a irrigação

Independentemente do método adotado para o sistema de irrigação, uma automação nestes sistemas, seja ele qual for só traz benefícios, pode-se pensar em praticidade, economia, tempo e aumento na qualidade do produto.

O objetivo é que um sistema automatizado possa monitorar a umidade do solo inibindo a irrigação ou na necessidade dela, atuar na quantidade suficiente funcionando durante um período pré-determinado ou dentro dos parâmetros necessários para cada tipo de cultivo.

De acordo com Naandanjain (2015), as principais vantagens da automatização são:

a) Diminuição de mão de obra;
b) Possibilita irrigações noturnas sem necessidades de acompanhamento;
c) Diminui a potência de acionamento;
d) Diminui custos de bombeamento;
e) Precisão nos tempos e turnos de irrigação;
f) Eficiência na aplicação da água;

Inseridos neste conceito, vários sistemas são oferecidos no mercado para automatizar o sistema de irrigação. A ideia deste trabalho é, além de automatizar, fornecer e armazenar informações que possam ser relevantes aos agricultores através de métodos e ferramentas computacionais.

3 A LINGUAGEM HTML

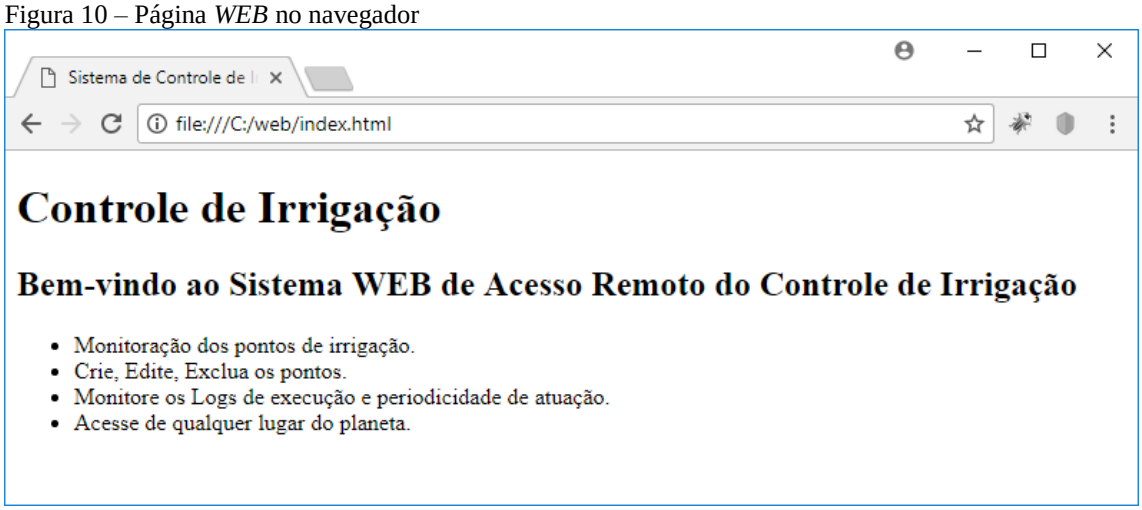
A Linguagem *Hypertext Markup Language* (HTML) foi desenvolvida no início da década de 90 e definida como um padrão de comunicação para apresentação de documentos na *Internet*. Atualmente está na versão 5, também conhecida como HTML5 com o padrão gerido pela *World Wide Web Consortium* (W3C), que segundo Eis, Ferreira (2016) é um consórcio internacional formado por organizações afiliadas representadas em várias nações, equipe em tempo integral, empresas, pesquisadores e o público, trabalham juntos para desenvolver padrões para a *WEB*. Eles tem a missão de conduzir a *World Wide Web* (WWW) para que atinja todo seu potencial desenvolvendo protocolos e diretrizes que garantam seu crescimento de longo prazo.

Todos os sites de *Internet* são desenvolvimento e dispostos em *Front-End* (interfaces), ou seja, as aplicações *WEB* que são acessadas através dos navegador, do Inglês *WEB Browser*, que são programas que interpretam as marcações criadas em HTML e as exibem na tela em formato de documentos. Os navegadores mais populares, são *Google Chrome*, *Microsoft Internet Explorer*, *Mozilla Firefox* para a plataforma *Windows* e *Safari* disponível mais em sistemas operacionais da *Apple* (DUCKETT, 2011).

O desenvolvimento de páginas em HTML se dá através da utilização das *tags*, elementos de marcação utilizados para inserir objetos e formatar a exibição de conteúdo na página. As *tags* são definidas como termos ou abreviações em inglês colocados entre colchetes angulares, como por exemplo <html>, que podem ou não possuir um correspondente para fechamento, </html>. Os clientes, conhecidos como *browsers* ou navegadores podem ler a HTML, interpretar e renderizar o conteúdo (W3C BRASIL, 2010).

Para que um documento seja exibido dentro dos padrões da W3C e seja exposto corretamente em um navegador, deve-se obedecer as estruturas das *tags* do HTML para que o documento fique formatado e organizado para os usuários. Como representação desta estrutura uma pequena amostra de um documento HTML pode ser observado conforme a Figura 10 que é uma demonstração de como uma página *WEB* é exibida no navegador.

Figura 10 – Página *WEB* no navegador



Fonte: O autor.

Para que esta página fosse exibida ao usuário com a formatação de título, subtítulo e com marcadores com uma lista de características do assunto, um arquivo em formato texto foi criado seguindo o padrão da linguagem estrutural do HTML.

Com o código salvo em um arquivo com o nome "*index*" com extensão "html", ficando "index.html", basta executá-lo para que o *browser* interprete as *tags* digitadas e represente o documento na tela (DUCKETT, 2011).

Para a linguagem HTML, tem como forma padrão a estrutura texto, com isso um simples editor de texto é o suficiente para criar uma página *WEB*, mas com a ampla divulgação de informações pela *WEB* e com a crescente demanda dos programadores, vários editores específicos foram desenvolvidos por grandes empresas de *software* e foram batizados de *Integrated Development Environment* (IDE). Alguns destes IDEs são gratuitos e outros comerciais. Alguns exemplos são, *ATOM* desenvolvido pelo *GitHub* sob a licença do MIT, *Notepad++* desenvolvido pela *Media Wiki*, RJ TextEd da empresa Embarcadero, dentre outros que são disponibilizados para uso gratuito. Alguns IDEs mais sofisticados são comercializados como por exemplo o *Dreamweaver* da *Adobe*, o *Codelobster* da empresa que leva o mesmo nome, *Eclipse* PDT da *Eclipse* e *Komodo* IDE da *ActiveState*.

4 A LINGUAGEM CSS

Desde que o HTML foi desenvolvido, ele apenas fornece a estrutura básica da página *WEB*, somente ele pode ser identificada pelo *browser* de navegação e para oferecer uma melhor aparência, foi desenvolvida e agregada ao HTML outra linguagem chamada *Cascading Style Sheets* (CSS), que é uma linguagem que define o *layout* de documentos HTML. Pode-se pensar em HTML como os ossos (estrutura) de uma página da *WEB* e o CSS como sua aparência (HARRIS, 2011).

A linguagem CSS é responsável por organizar a forma como elementos HTML são apresentados nos monitores, celulares, *tablets*, TVs, projetores e até mesmo quando se imprime uma página HTML, o CSS está envolvido. Pode-se chamar de responsividade de uma página *WEB*, ou seja, a capacidade desta página se adaptar a diferentes dispositivos, com diferentes resoluções e formatos (SCUDERO, 2016).

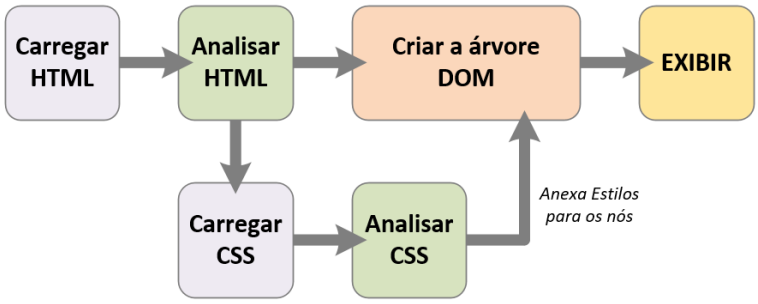
A partir do desenvolvimento desta linguagem, os documentos HTML, além de ganhar mais cores e estilo, obteve um grande aliado na parte de organização. O CSS deixa o código mais limpo, facilitando o entendimento por parte de quem programa e também por parte do navegador que otimiza a renderização e apresentação da página *WEB* (HARRIS, 2011).

4.1 Funcionamento do CSS

O navegador é quem processa toda a informação contida no documento HTML criado para a página. Ele faz a leitura desta estrutura e associa o conteúdo do documento com a informação de estilo definida no CSS. Este processo é executado em duas etapas, segundo Mozilla (2016):

a) O navegador converte o HTML e CSS para dentro do Modelo de Objeto do Documento (DOM). O DOM representa o documento na memória do computador. Ele combina o conteúdo do documento com o seu estilo.
b) O navegador mostra o conteúdo do DOM, conforme a Figura 11.

Figura 11 – Fluxo HTML e CSS



Fonte: Adaptado de (Mozilla, 2016).

Um DOM tem uma estrutura semelhante a uma árvore. Cada elemento, atributo e trecho de texto na linguagem, torna-se um DOM, um nó na estrutura desta árvore, desta forma todo o conteúdo exibido pode ser manipulado na sua representação (MOZILLA, 2016).

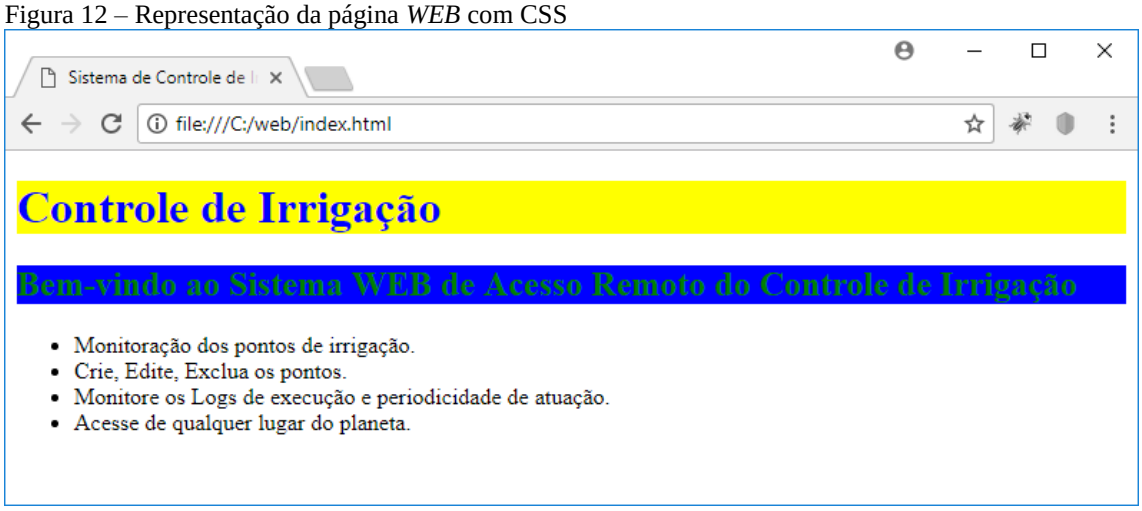
4.2 CSS aplicado ao HTML

Para que a linguagem CSS seja aplicada ao HTML, alguns métodos de inserção do código são permitidas para o seu correto funcionamento. Ela pode ser externa ao código HTML, escrito em um arquivo a parte, ou seja, separado do código HTML, mas com referência a este arquivo, é considerado a melhor forma de se trabalhar com o CSS ao HTML. Outro método é interno ao código HTML, que é quando o código CSS faz parte do mesmo arquivo HTML, ele será inserido dentro das *tags* do HTML. Este procedimento pode ser útil em algumas circunstâncias (exemplo do caso de se estar trabalhando com um sistema de gerenciamento de conteúdo no qual não possa modificar os arquivos CSS diretamente), mas não é tão eficiente quanto as folhas de estilo externas. Em um site, o CSS precisaria ser repetido em todas as páginas e atualizado em vários lugares se as alterações forem necessárias e o terceiro método é o estilo *Inline*, ou seja, a folha de estilo seria de uso exclusivo a apenas uma das páginas HTML.

Independentemente do método a ser utilizado na codificação CSS, a exibição da página será a mesma. O que muda é a organização no desenvolvimento do código HTML juntamente com o CSS e a facilidade na necessidade de manutenção do sistema, principalmente por outro programador que não desenvolveu a página *WEB*.

A página *WEB* representada no DOM, passa a obedecer as características determinadas pelo desenvolvedor onde os estilos são aplicados e recebem definições distintas e no navegador ambas são exibidas conforme a Figura 12.

Figura 12 – Representação da página *WEB* com CSS



Fonte: O autor.

5 A LINGUAGEM *JAVASCRIPT*

A linguagem denominada *JavaScript* é uma linguagem de programação interpretada que traz uma funcionalidade dinâmica para as páginas *WEB*, com ela é possível mostrar informações aos usuários através da passagem do *mouse* sobre um item dentro da tela do navegador, ou mostrar um novo texto, cores, imagens, validação de campos a criação de menus, presentes na tela e que compõem o conteúdo, sendo possível até arrastar um objeto para um novo local, todas essas coisas são feitas através do *JavaScript* (NIXON, 2015).

JavaScript é muito versátil, enriquecendo as funcionalidades do HTML expostas aos usuários, fornecendo desde simples carrosséis, galerias de imagens, *layouts* flutuantes e respostas a cliques de botão a criação de jogos, gráficos 2D e 3D animados e aplicativos abrangentes baseados em bancos de dados (MOZILLA, 2018).

As versões mais avançadas da linguagem *JavaScript*, permitem adicionar mais funcionalidades a um site, dentro de um navegador, o *JavaScript* pode ser conectado aos objetos de seu ambiente para fornecer controle programático sobre eles, contendo uma biblioteca padrão de objetos, como matrizes, datas e expressões matemáticas, e um conjunto principal de elementos de linguagem, como operadores, estruturas de controle e instruções, enfim, o *JavaScript* pode ser estendido para uma variedade de finalidades, complementando-o com objetos adicionais:

a) O *JavaScript Client-Side* fornece objetos para controlar um navegador e seu DOM e permitem que um aplicativo coloque elementos em um formulário HTML e responda a eventos do usuário, como cliques do mouse, entrada de formulário e navegação de página.
b) O *JavaScript Server-Side,* o *Node.js* fornece objetos relevantes para executar o *JavaScript* e permitem que um aplicativo se comunique com um banco de dados, forneça continuidade de informações de uma chamada para outra do aplicativo ou execute manipulações de arquivos em um servidor.

No navegador, o *JavaScript* pode alterar a aparência da página da *WEB* (DOM) e da mesma forma no servidor pode responder a solicitações personalizadas de código escrito no navegador.

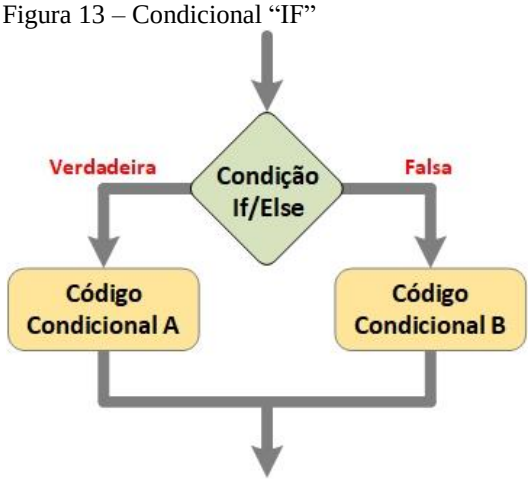
5.1 Funcionamento da linguagem *JavaScript*

A linguagem *JavaScript* tem semelhança com as sintaxe da linguagem de programação C. Ela faz uso de variáveis com controle de conteúdo que aceitam apenas determinados tipos

de dados como, *string*, caracteres, numerais inteiros, booleanos e fracionais, além de vetores e matrizes. Outras características são, o uso de operadores para efetuar cálculos aritméticos como, soma, subtração, divisão e multiplicação, e outras funções matemáticas para chegar a um módulo inteiro de uma divisão. Operadores de comparação são geralmente usados dentro de uma construção, ou seja, uma instrução onde você precisa comparar dois itens para uma determinada tomada de decisão. Os operadores lógicos são usados para criar operações mais complexas, combinando condições simples. O valor de uma expressão lógica é ou verdadeiro ou falso.

Para a execução das declarações em ordem linear tem-se duas opções, seja ela linear ou a forma condicional, a primeira opção não é praticada por questão de boas práticas e interpretação de código, com isso a forma condicional é praticada, onde se escolhe entre dois caminhos distintos baseados em um valor booleano, de acordo com a Figura 13 (WILTON; McPEAK, 2010).

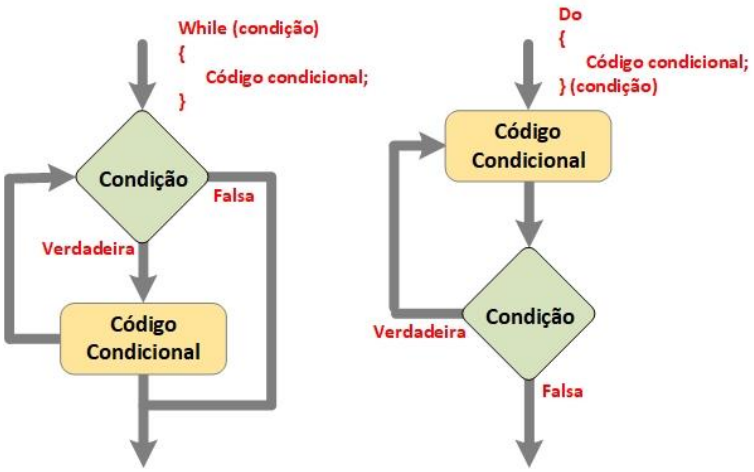
Figura 13 – Condicional “IF”



Fonte: Adaptado de (Wilton; McPeak, 2010, p. 51).

Da mesma forma como o valor é tratado com o “*if / else”* é possível permanecer dentro de um trecho de código até que uma condição seja atingida, conforme a Figura 14. Para o *JavaScript* os *loops “While”* e “*Do”* fazem esta tratativa. O “*While”* executa um trecho alternadamente até que o valor em questão seja verdadeiro, já o comando “*Do”* executa pelo menos uma vez para depois analisar a condição da tratativa.

Figura 14 – Condicional "*While" e "Do"*

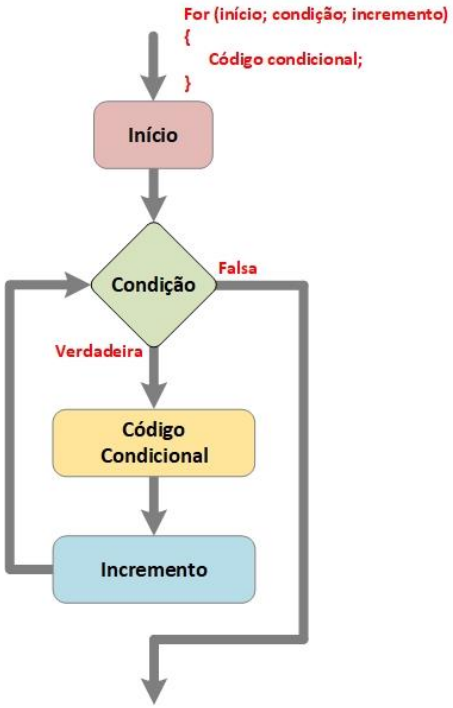


Fonte: Adaptado de (Wilton; McPeak, 2010, p. 76).

Com os *loops "While"* e "*Do"*, eles se baseiam em condições booleanas para definir quando será a hora de partir para outra determinada parte do código ou simplesmente parar uma execução. Quando se precisa fazer um *loop* em determinado trecho do algoritmo porém com uma determinada quantidade de vezes definidas, utilização a estrutura "*For"*, conforme a Figura 15 (WILTON; McPEAK, 2010).

A estrutura de *loop "For"* trabalha com três pontos de controle que são o ponto inicial, a condição para que saída o ponto final da execução e a parte incremental para cada vez que o código é executado.

Figura 15 – *Loop* condicional "*For"*

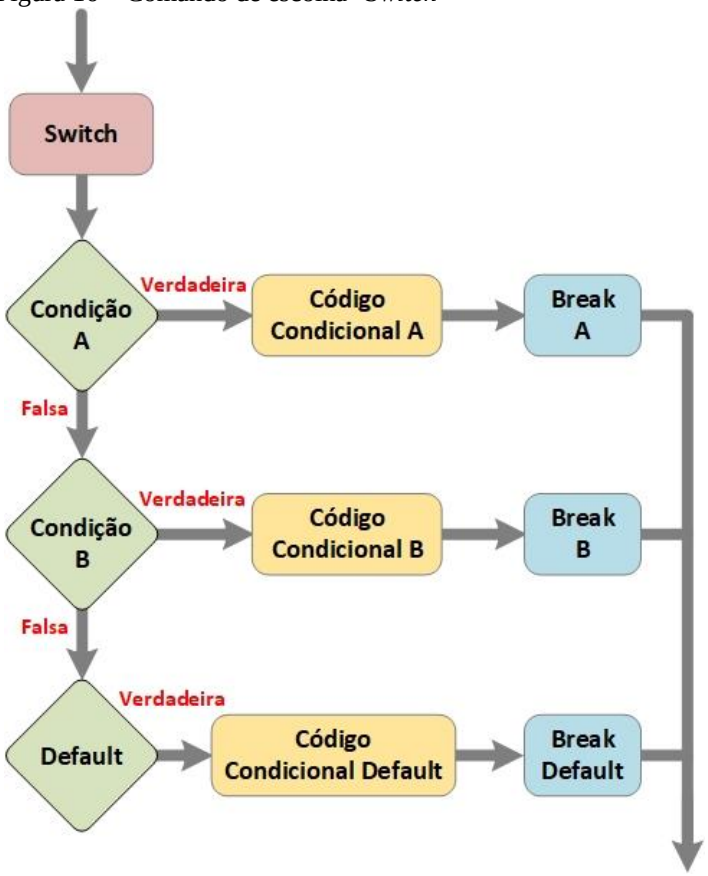


Fonte: Adaptado de (Wilton; McPeak, 2010, p. 71).

Outro comando da linguagem *JavaScript* é o "*Switch"*, ele funciona da mesma forma que o comando "*If"*, mas de uma forma mais direta. Com este comando os códigos que possuem uma variável que possa assumir uma extensa gama de resultados, facilita a interpretação e melhora o desempenho do código.

O "*Switch"* recebe por parâmetro o valor que será verificado. Os "*Case"* que acompanham, informa-se um valor que será comparado e caso verdadeiro ele executa um determinado trecho de código. O "*Break"* é colocada posteriormente ao código executado, que confirma que foi realizado e que a aplicação irá sair do laço "*Switch"* e que outros "*Cases"* não serão testado. Para finalizar usa-se o comando "*Default"* que se nenhum dos "*Cases"* atendeu a necessidade do código, ele assume como execução padrão, ou seja, ele sempre será executado caso nenhuma condição tenha sido reproduzida, conforme a Figura 16 (WILTON; McPEAK, 2010).

Figura 16 – Comando de escolha "*Switch*"



Fonte: Adaptado de (Wilton; McPeak, 2010, p. 67).

Todas estas características de se poder manipular valores e atribuições dentro de uma aplicação *WEB*, propicia o que chamamos de dinamismo em uma página *WEB* (NIXON, 2015).

Outras funções de manipulação do conteúdo oferecido para o usuário final favorece e enriquece a apresentação. Estas funções são criadas para permitir o reaproveitamento de código já construído (pode ser pelo programador da página *WEB* ou adquirido de outros programadores). Com isso evita-se que um trecho de código seja repetido dentro de um mesmo algoritmo, com isso a alteração e manutenção do código, caso necessário, se dá em apenas na função, facilitando a programação e deixando o código mais limpo (PINHO, 2018).

As funções são chamadas através de eventos que ocorrem durante a navegação do usuário na página *WEB* e são divididas e interpretadas conforme a necessidade e ocasião da programação pré-definidas pela linguagem *JavaScript*, o programador tem uma grande variedade de opções para trabalhar as manifestações do usuário na navegação da página *WEB*.

6 O *BOOTSTRAP*

Bootstrap é um *framework Front-End* disponível de forma gratuita para um desenvolvimento *WEB* rápido e fácil, nele estão inclusos as linguagens HTML, CSS que inclui modelos baseados para tipografia, formulários, botões, tabelas, navegação, modais, carroceis de imagens e muitos outros *plug-ins JavaScript* opcionais (W3SCHOOL, 2016).

O *Bootstrap* também oferece a capacidade de criar facilmente designs responsivos, que segundo Silva (2015), é uma característica que demonstra que um site pode ser visto de diversas formas e em diversos contextos, e é para isto que os sites devem estar preparados. O design responsivo, como o próprio nome já indica, consegue responder ao tamanho da tela para se adequar da melhor forma. Ao invés de criar várias versões de sites separados, um para *mobile* (*smartphones* e *tablets*) e um para *desktops*, como também é possível fazer, você faz apenas um site que vai se adaptar muito bem a qualquer tela em que ele for carregado.

Desta forma o *Bootstrap* mostra uma preocupação e atingir qualquer dispositivo para levar a informação, além de uma padronização.

Na elaboração deste trabalho o *Bootstrap* estava na versão 4.0.

6.1 O *Bootstrap* e a *WEB*

O *Bootstrap* é mantido por uma pequena equipe de desenvolvedores no *GitHub*, que segundo Schmitz (2015), é um serviço *WEB* que oferece diversas funcionalidades extras onde programadores podem usar gratuitamente para hospedar projetos pessoais, além de quase todos os projetos, *frameworks* e bibliotecas sobre desenvolvimento *Open-Source* estão disponíveis, com o *Github* é possível acompanhar as novas versões, contribuir informando *bugs* ou até mesmo enviando código e correções afirma Schmitz (2015).

Todas as funcionalidades do *Github* são aplicadas ao *Git*, que é um sistema de controle de versão de arquivos. Através dele pode-se desenvolver projetos na qual diversas pessoas podem contribuir de forma simultânea no mesmo, editando e criando novos arquivos sem correr o risco de suas alterações serem sobrescritas (SCHMITZ, 2015).

O *Bootstrap* mantém um padrão de objetos que estão disponíveis para utilização de forma *Open-Source*. Ele engloba todas as *tags* do HTML, estilização do CSS e funções do *JavaScript*, porém de forma consolidada.

Os modelos de *layout* (*containers, grids, media objects*, utilitários de *layout*), conteúdos (tipografias, imagens, tabelas e figuras), Componentes (alertas, destaques, caminhos de páginas

abertas, botões, grupo de botões, cartões, carrossel, objetos ocultos, cortina de informações ocultas, formulários, grupos de informações de entrada, títulos jumbo, grupo de listas, modal, *menu* nav, *menu* navbar, paginação, janelas *popup*, barra de progresso, posicionamento no texto, informação nos botões) e Utilidades (bordas, efeitos sobre botões, ícones, cores, embutidos, substituição de imagens, posicionamento de objetos, sombreamentos, dimensões, espaçamentos, textos, alinhamentos e visibilidade) (SCHMITZ, 2015).

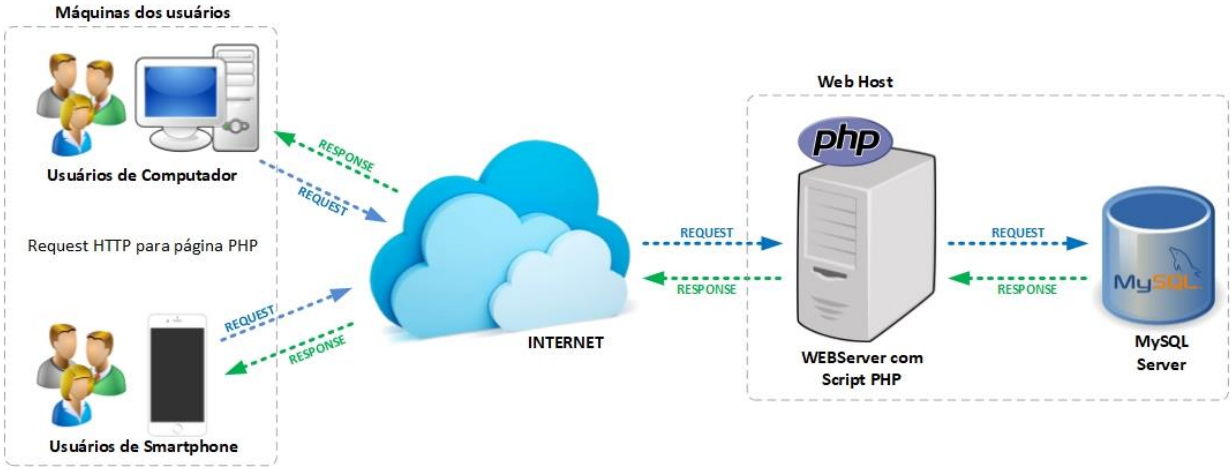
7 A LINGUAGEM PHP E MYSQL

O *Hypertext Preprocessor* (PHP) é uma linguagem de *script* que pode ser utilizada no desenvolvimento de aplicações que atendem a qualquer demanda no ramo de Tecnologia da Informação (TI). Segundo Converge, Morgan, Park (2004), o PHP é um módulo oficial do servidor HTTP *Apache*, o líder do mercado de servidores *WEB* livres que constitui aproximadamente 55 por cento da *World Wide Web*. Isso significa que o mecanismo de *Script* do PHP pode ser construído no próprio servidor *WEB*, tornando a manipulação de dados mais rápida. Assim como o servidor *Apache*, o PHP é compatível com várias plataformas, o que significa que ele executa em seu formato original em várias versões de sistemas operacionais como o *Windows*, *Linux*, *FreeBSD*, *Mac OS*, *Novell Netware*, *RISC OS*, *AIX*, *IRIX* e *Solaris*. Todos os projetos sob a égide da *Apache Software Foundation* – incluindo o PHP – são software de código-fonte aberto, ou seja, de livre utilização. Algumas ferramentas disponíveis no mercado de desenvolvimento e que são concorrentes ao PHP são as linguagens ASP.Net da empresa *Microsoft*, *ColdFusion* da empresa *Adobe Systems* e *Java Server Pages* (JSP) da empresa *Sun Microsystems*.

7.1 Funcionamento do PHP

Os *scripts* PHP para oferecer o dinamismo nas páginas desenvolvidas com a sua tecnologia, todo o processamento é definido do lado do servidor, ou seja, serviço oferecido pelo servidor da aplicação onde a página *WEB* está hospedada, conforme a Figura 17. O usuário solicita o *link* ou verificação de autenticação através das páginas *WEB* oferecida pelo *Browser* em seu computador ou *smartphone* pessoal e estando este conectado na *Internet*, a requisição (*REQUEST*) é solicitado ao *WEBServer* que processa através do *script* PHP e responde (*RESPONSE*) para o usuário em HTML dinâmica.

Figura 17 – Estrutura cliente/servidor PHP



Fonte: Adaptado de (Converge; Morgan; Park, 2004, p. 27)

Até a elaboração deste trabalho a linguagem PHP foi registrada na sua versão 7.2.3 onde após muitos anos de trabalho no seu desenvolvimento os desenvolvedores aprovaram os benefícios trazidos por esta nova compilação com uma grande performance. Esta nova versão não trouxe apenas melhorias em performance, mas também novas funcionalidades, além de implementar e fortificar novos recursos na orientação a objetos, afirma Beraldo (2015).

Para a linguagem PHP ser interpretada e tornar-se parte o HTML, ela deve ser executada em um servidor que reconheça a sua estrutura. De acordo com Converge, Morgan e Park (2004), o PHP é um módulo oficial do *Apache HTTP Server*, um servidor *WEB* gratuito líder de mercado que atende cerca de 67 por cento da *World Wide Web* (de acordo com o amplamente citado *Netcraft WEB*). Isso significa que o mecanismo de *script* PHP pode ser construído no servidor da *WEB* em si, levando a um processamento mais rápido, a uma alocação de memória mais eficiente e simplificada manutenção. Como o Servidor *Apache*, o PHP é totalmente multi-plataforma, o que significa que ele é executado no *Unix*, bem como no *Windows* e também no *Mac OS X*.

O *script* desenvolvido em PHP seja executado em *Server-Side*, ou seja, tudo é executado do lado do servidor e as informações são devolvidas ao usuário em formato HTML para que possam ser exibidas no navegador.

O programador ao desenvolver uma página *WEB* ele deve ter conhecimentos de funções e métodos, para tomadas de decisões no *script*, bem como armazenar temporariamente informações que não possam ser executadas em variáveis (DAVIS; PHILLIPS, 2007).

O PHP, assim como outras linguagens de programação, possui a capacidade de realizar cálculos matemáticos e utilização de operadores relacionais para comparação entre dois

operadores para tomada de decisão no comportamento do algoritmo. Da mesma forma, a utilização dos operadores lógicos, condicionais e de repetições.

7.2 PHP orientado a objetos

A programação orientada a objetos possui os mesmos objetivos das funções, principalmente para tornar o código de reutilização mais dinâmico e fácil. Ele utiliza classes para agrupar funções e variáveis juntas como um único objeto (NIXON, 2015). Pode comparar a programação orientada a objetos como caixas pretas que podem funcionar sem que se saiba exatamente em detalhes como o processo é realizado. As mesmas características das funções são usadas, mas recebem um novo nome ao serem definidas em classes, denominado Métodos. Quando se cria um novo objeto de uma classe, ele é chamado de instância dessa classe. Qualquer variáveis definidas na classe obtêm espaço de armazenamento separado em cada instância, afirma Nixon (2015).

As classes geralmente são criadas e armazenadas em arquivos separados para serem reutilizadas.

A construção da classe define-se primeiramente o seu nome, que no caso do exemplo SS, a classe foi denominada Cliente, seguindo as mesmas regras de nomenclatura de variáveis e funções. O código que compõe a classe é colocado entre chaves.

Para a definição dos atributos, que segundo Nixon (2015), são as características das classes que modificam o seu estado.

Os métodos das classes são as funções que são definidas dentro da classe. Estes métodos trabalham dentro do escopo da classe, incluindo suas variáveis. Quando se deseja utilizar uma variável usa-se um método especial chamado de construtor que é chamado quando uma nova instância de uma classe é criada para realizar a sua função, que inicializa a classe para definir seus valores. O construtor é definido criando um método que tenha o mesmo nome que a classe.

O método construtor, definido pelo programador, é sempre instanciado instancia um objeto da classe. Ele é considerando uma função especial do PHP que aceitam passagens de parâmetros e o método construtor define os valores iniciais dos atributos de um objeto (NORDBOTTEN, 2009).

Quando se cria um objeto, está se criando uma instância de uma classe, o que significa estar instanciando uma classe. A nova construção instancia uma classe alocando memória para esse novo objeto, o que significa que requer um único argumento, que é uma chamada para um construtor. O nome do construtor fornece o nome da classe para instanciar e o construtor

inicializa o novo objeto. A nova construção retorna uma referência ao objeto que foi criado. Esta referência é geralmente atribuído a uma variável.

7.3 O banco de dados MySQL

O banco de dados MySQL é um banco de dados relacional de código fonte aberto utilizado pela maioria dos desenvolvedores e programadores *WEB*, devido a sua facilidade de uso e robustez, além de ser extremamente seguro. Possui uma grande comunidade de usuários que contribuem para o seu sucesso. Segundo Converge, Morgan e Park (2004), o MySQL foi projetado para ser fácil de usar e sua escalabilidade e flexibilidade o tornam adequado para praticamente qualquer aplicação. Seus principais concorrentes são *Oracle, Sybase, PostgreSQL, InterBase, SQLite, MSSQL, Firebird*, dentre outras. Assim como o PHP, o MySQL tem portabilidade para os mesmos sistemas operacionais.

A forma como os dados são manipulados na base de dados, ou seja, inseridos, editados, excluídos e pesquisados, utiliza-se a *Structured Query Language* (SQL), que é uma linguagem padrão universal para manipular bancos de dados relacionais através dos Sistema de Gerenciamento de Banco de Dados (SGBD). Logo, saber o que é SQL e como utilizá-la é fundamental para qualquer desenvolvedor de *softwares*, afirma Beighley (2012).

Para o desenvolvimento de uma aplicação para controle e monitoração de um sistema de irrigação está se utilizando de informações que não possam estar disponíveis para qualquer pessoal. Com o *script* PHP através das suas ferramentas pensou-se em criar uma estrutura inicial a página *WEB*, com controle de usuários através de nome de acesso e senha. Estes usuários estar-se-á cadastrados na base de dados do MySQL. Desta forma o acesso do usuário ao sistema é auditado pelo *script* PHP que verifica se o usuário está cadastrado no sistema e qual o seu nível de acesso, podendo limitar informações ou ter acesso ilimitado.

O MySQL é uma linguagem de manipulação de dados que possui seu próprio ambiente de desenvolvimento, permitindo que que mova-se dados e altere configuração do banco de dados. Quando se cria uma nova tabela para armazenamento de informações no banco de dados deve-se usar um usuário e senha para o acesso. Atribuindo usuários do banco de dados permite limitar o acesso a tabelas (NORDBOTTEN, 2009).

Os servidores MySQL podem hospedar vários bancos de dados, atendendo a páginas *WEB* de diferentes programadores. Uma aplicação *WEB* pode usar seu banco de dados próprio proprietário ou um banco de dados padrão como o MySQL. Você pode ter instalado o MySQL sozinho ou ter acesso a ele através do seu *Internet Service Provider* (ISP). A maioria ISPs que

suportam PHP também fornecem um banco de dados MySQL para seu uso. Para hospedar uma página *WEB* a um ISP e fazer uso de um banco de dados, é necessário o endereço *Internet Protocol* (IP), o nome definido para o banco de dados, o nome de usuário e a senha de acesso (LURIG, 2008).

7.4 SQL linguagem de consulta estruturada

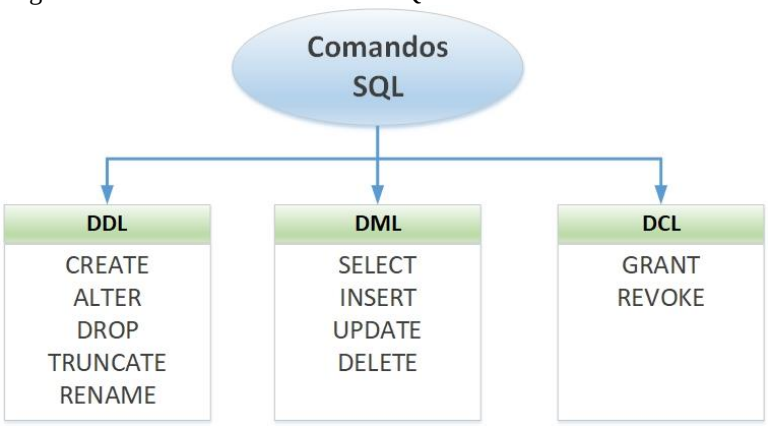
O Banco de Dados (BD) nos fornece inúmeras opções para utilizar a sua estrutura como base para o funcionamento de aplicações, sejam elas no padrão *WEB* ou ambiente operacional. Com ele é possível fazer cadastro de usuários, produtos, serviços e armazenar opções de usuários. Para a manipulação destes cadastros, seja para a inclusão, edição, pesquisa e exclusão de informações, utiliza-se a SQL, uma linguagem de uso universal padrão *American National Standard Institute* (ANSI) que atende a todas as demandas de relacionamentos com o banco de dados na aplicação (HALVORSEN, 2016).

Os dados armazenados em um Sistema de Gerenciamento de Banco de Dados Relacional (RDBMS), que é a base do SQL, são armazenados em objetos de banco de dados que são chamados como tabela. Essa tabela é basicamente uma coleção de entradas de dados relacionados e consiste em várias colunas e linhas, onde as colunas são os campos da tabela e as linhas os registros. Esta é a forma mais comum e mais simples de armazenamento de dados em um ambiente relacional afirma Halvorsen (2016).

A SQL tem seu funcionamento baseado em "*Query*", que no português quer dizer "Consultar", divididas em subconjuntos para melhor satisfazer a necessidade nas operações. Estes subconjuntos são o DDL (*Data Definition Language* ou Linguagem de Definição de Dados), o DML (*Data Manipulation Language* ou Linguagem de Manipulação de Dados) e o DCL (*Data Control Language* ou Linguagem de Controle de Dados (HALVORSEN, 2016).

A estrutura destes subconjuntos são apresentados conforma a Figura 18.

Figura 18 – Estrutura de comandos SQL



Fonte: Adaptado de (Halvorsen, 2016, p. 7)

Quando se pensa em um sistema que irá fazer uso de bando de dados, logo já se imagina a criação das tabelas e como os dados serão armazenados. O que a maioria dos programadores não se preocupam é como o sistema irá se comportar com a interação com o banco de dados. Esta interação quer dizer organização das tabelas, desempenho entre tabelas, padronização dos dados, normas e regras. Tudo isso só se consegue com um banco de dados relacional (DIAS, 2017).

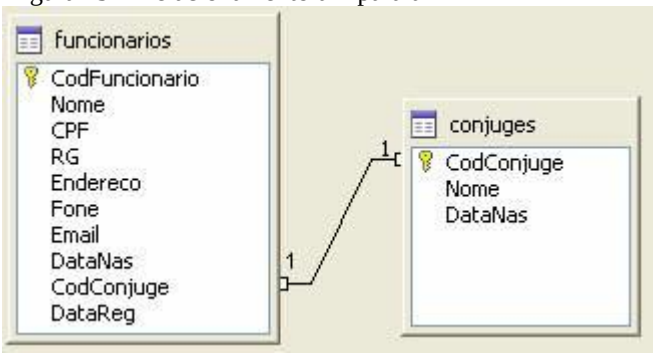
De acordo com o professor Fais (2009), um banco de dados relacional é um conjunto de tabelas relacionadas entre si gerenciadas por um SGBD. Dados relacionados são armazenados em tabelas separadas e permite que sejam identificados usando uma chave comum para ambas as tabelas. A chave é a relação entre as tabelas. A seleção de uma chave primária é uma das decisões mais importantes que você fará na criação de um novo banco de dados e deve-se garantir que esta chave selecionada seja única. Incluindo campos-chave de outra tabela para formatar o *link* entre as tabelas é chamado de relação de chave estrangeira, como um produto para os compradores ou um comprador para um salário (DAVIS; PHILLIPS, 2007).

Quando se estrutura um projeto de banco de dados, criasse as tabelas para os registros, deve-se proceder identificando os tipos de dados para se definir os tipos de relacionamentos. As Relações dos bancos de dados são quantificadas com as categorias “Um para Um”, “Um para Muitos” e “Muito para Muitos”.

Relacionamento Um para Um é o tipo que se dá de forma direta entre duas tabelas, quando a chave primária do registro de uma determinada tabela pode ser utilizada uma única vez em um dos registros da outra tabela. No exemplo da Figura 19, tem-se duas tabelas, uma para cadastro de funcionários e outra para cadastro de cônjuges (esposa ou marido), sendo este, um típico exemplo de relacionamento um para um, pois neste caso, o código de cada cônjuge poderá ser especificada uma única vez na tabela de funcionários, visto que para cada

funcionário existirá apenas um cônjuge. Este tipo de relacionamento é pouco utilizado, sendo que se existe apenas um relacionamento entre as tabelas, todos os dados podem fazer parte de uma mesma tabela (FAIS, 2009).

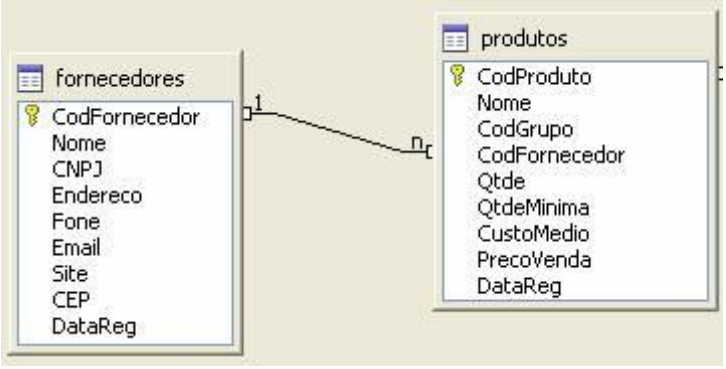
Figura 19 – Relacionamento um para um



Fonte: (FAIS, 2009).

Relacionamento Um para Muitos é um tipo de relacionamento que também acontece de forma direta entre duas tabelas sempre que a chave primária do registro de uma determinada tabela é utilizada várias vezes em outra tabela, sendo este, o tipo de relacionamento mais comum entre tabelas de um banco de dados relacional. O exemplo da Figura 20 demonstra a relação entre uma tabela para cadastro de produtos e uma tabela para cadastro de fornecedores, onde um mesmo fornecedor vende vários produtos, podendo o seu código ser informado várias vezes em diferentes registros da tabela de produtos (FAIS, 2009).

Figura 20 – Relacionamento um para muitos



Fonte: (FAIS, 2009).

Relacionamento Muitos para Muitos é um tipo de relacionamento que acontece de forma indireta entre duas tabelas, pois para que ele possa ser concebido é necessário a geração de uma terceira tabela. Na prática o relacionamento Muitos para Muitos não existe de fato, o que existe é dois ou mais relacionamentos Um para Muitos, que ganha o sentido de Muitos para Muitos.

Ocorre sempre que surge a necessidade de se relacionar duas chaves primárias de registros de diferentes tabelas em vários registros de uma terceira tabela. O exemplo da Figura 21 considera um sistema em que o cliente de uma empresa possa fazer reserva de produtos para serem comprados. Neste caso, tem-se uma tabela para cadastro de produtos, uma tabela para cadastro de clientes e uma tabela para registro de reservas. Observe que na tabela para registro de reservas, um mesmo cliente pode fazer reserva de vários produtos e um mesmo produto pode ser reservado por vários clientes. Com isso, surgem duas relações Um para Muitos, que ganha o sentido de Muitos para Muitos (FAIS, 2009).

Figura 21 – Relacionamento muitos para muitos

Fonte: (FAIS, 2009).

8 MICROCONTROLADOR ARDUINO

O projeto Arduino surgiu em meados do ano de 2005 como um projeto para estudantes de uma escola Italiana. Originalmente ele foi desenvolvido como um recurso para auxiliar os estudantes no ensino e com o seu sucesso ele foi comercialmente lançado por dois pesquisadores que acreditaram neste projeto, Massimo Banzi e David Cuartielles, tornando-se um produto de sucesso entre fabricantes e estudantes devido a sua fácil utilização e a durabilidade que ele proporciona (MONK, 2013).

O Arduino tem a estrutura de *hardware Open-Source*, com uma infinidade de possibilidades para a criação de dispositivos que permitam interação com o ambiente externo, dispositivos estes que utilizem como entrada sensores de temperatura, aproximação, velocidades, inclinação, luz, som etc., e como saída ele possui portas de nível lógico analógico e digital para acender *Light Emitting Diode* (LEDS), acionar motores, indicações em *displays*, etc.

A plataforma original, já contemplava uma quantidade de recursos de grande utilidade, mesmo com restrições, possuía grandes possibilidades com seis entradas analógicas, 1 *Universal Asynchronous Receiver Transmitter* (UART), *Inter-Integrated Circuit* (I2C), *Serial Peripheral Interface* (SPI) e 6 *Pulse Width Modulation* (PWM). Hoje, na mesma linha tem-se modelos que permitem até 16 entradas analógicas, 14 PWMs, 4 UARTs, e com memória comparável a plataformas complexas como a família *Advanced RISC Machine* (ARM) (ARDUINO.CC, 2017).

O Arduino possui terminais de entrada e saída, por onde deve-se conectar os dispositivos capazes de interpretar o ambiente externo e representar ao processamento do microprocessador. Da mesma forma, o Arduino é capaz de controlar motores ou outras saídas físicas conectadas ao seu terminal de saída, como pode ser visto na Figura 22.

Figura 22 – Arquitetura Arduino

Fonte: O autor.

A placa comercializada é formada por vários componentes que trabalham em conjunto para formar o microcontrolador. Em geral são formadas por microprocessadores *Atmel* que segundo Tawil (2016), é um microcontrolador *Reduced Instruction Set Computer* (RISC) que

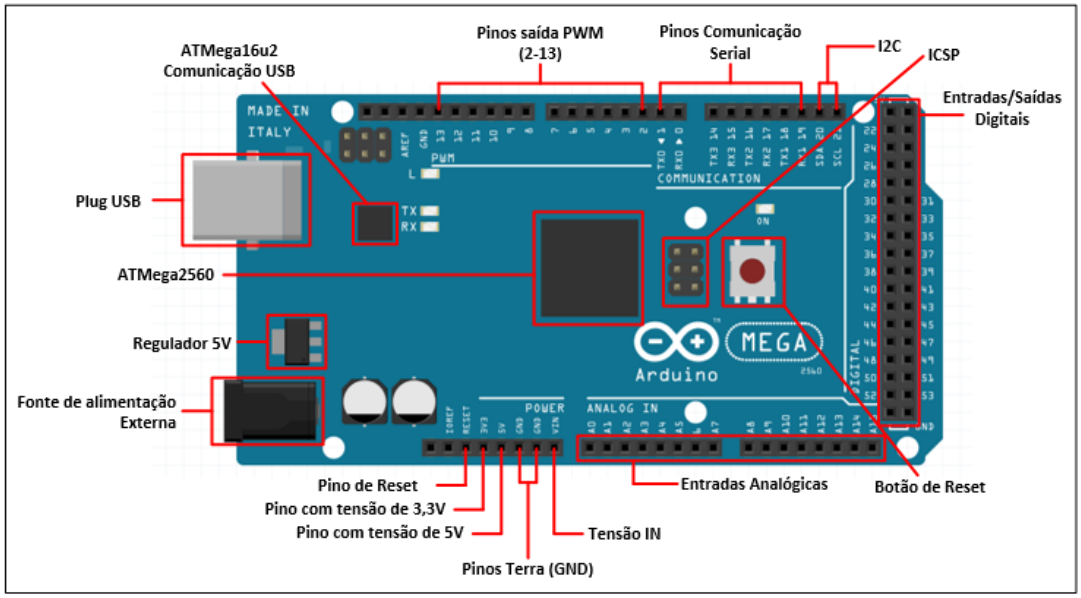
possui apenas um único chip de arquitetura *Harvard* de 8 *bits*. Foi o pioneiro a utilizar uma memória *flash* para armazenar o código de programação, contrariando os seus concorrentes que ainda utilizavam memórias do tipo Programmable Read Only Memory (*PROM), Erasable Programmable Read Only Memory (EPROM) ou Eletrical Erasable Programmable Read Only Memory (EEPROM)* (TAWIL, 2016).

Os outros componentes que agrupam esta formação são, o cristal ou oscilador, regulador de tensão, botão de *reset*, um plugue de alimentação, pinos conectores, e alguns LED para indicar ao programador usuário a verificação do funcionamento, uma porta *Universal Serial Bus* (USB) ou Serial para conecta-lo ao *Personal Computer* (PC) e realizar o *upload* do código de programação (BAYLE, 2013).

Existem no mercado uma grande variedade de modelos de microcontroladores Arduino que se diferenciam no tipo de controlador, na quantidade de pinos, no tamanho da memória e na frequência de trabalho.

O Arduino *MEGA* 2560, demonstrado na Figura 23, foi projetado para projetos mais elaborados, possui 54 pinos de entrada e saídas digitais, 16 entradas analógicas. É uma placa de microcontrolador baseada no *ATmega2560*. Possui 54 pinos de entrada / saída digitais (dos quais 15 podem ser usados como saídas PWM), 16 entradas analógicas, 4 Portas Seriais de *hardware* (UART), um oscilador de cristal de 16 MHz, 256 k*Bytes* de memória *Flash* sendo 8 k*Bytes* usados pelo *Bootloader*, 8 k*Bytes* de memória SRAM, 4 k*Bytes* de EEPROM, uma conexão USB, conector de alimentação, um conector *In-Circuit Serial Programming* (ICSP), e um botão de *reset*.

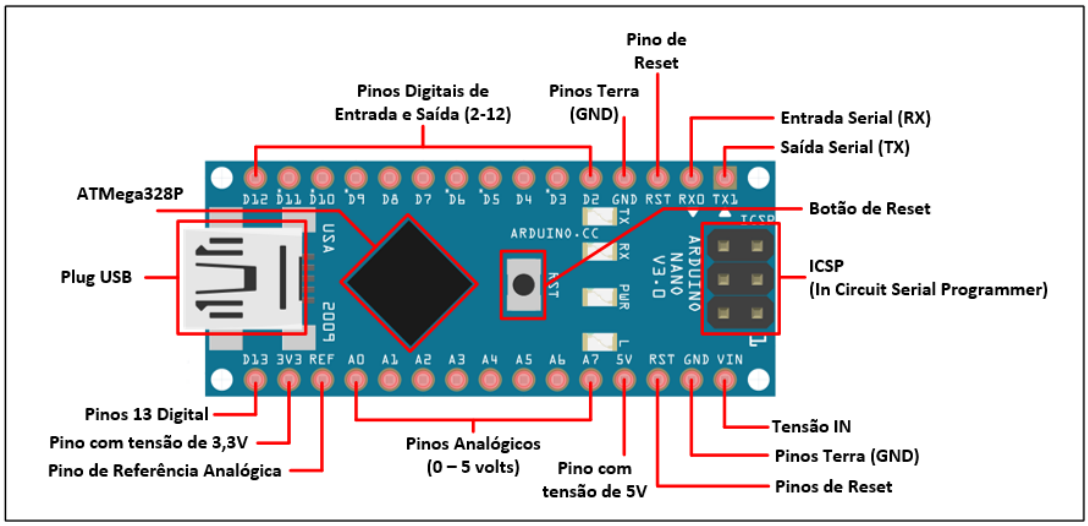
Figura 23 – Microcontrolador Arduino *MEGA* 2560



Fonte: Adaptado de (SOUZA, 2016).

O Arduino *NANO*, conforme a Figura 24, é similar ao modelo UNO. O Arduino *NANO* não possui *plug* de alimentação, mas pode ser alimentado através da conexão USB padrão Mini-B, fonte de alimentação externa não regulada de 6-20 *Volts*, através do pino 30 ou fonte de alimentação externa regulada de 5 *Volts* através do pino 27. A fonte de energia é automaticamente selecionada para a fonte de tensão mais alta quando necessário. Este Arduino possui 14 pinos de entradas/saídas digitais, sendo os pinos 3, 5, 6, 9, 10 e 11 são utilizados para PWM, 8 pinos analógicos de 10 *bits* de resolução, 32 kB de memória *Flash*, sendo desta 2 kB usado para o *bootloader*, 2 kB de *Static Random Access Memory (SRAM)* e 1 kB de *EEPROM*.

Figura 24 – Microcontrolador Arduino *NANO*



Fonte: Adaptado de (CIRCUITS TODAY, 2017).

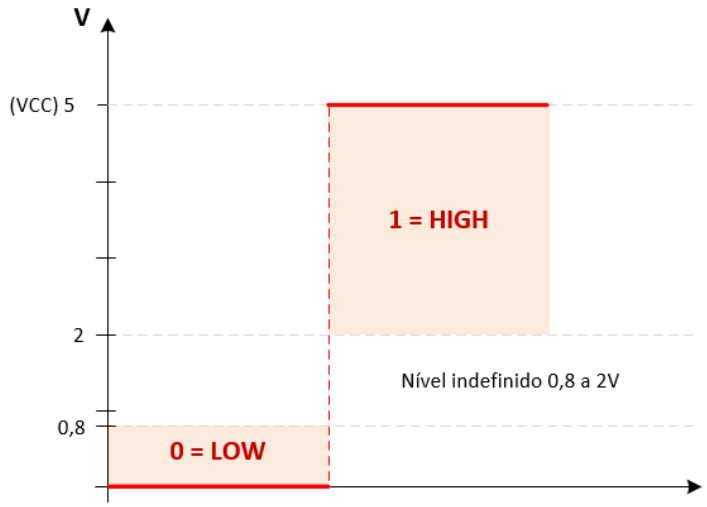
Dentre os principais elementos de conexão e trabalho do Arduino, estão as portas de comunicação. É por elas que o Arduino faz a comunicação de entrada e saída para o mundo externo, lendo os sinais gerados por sensores ou comandando atuadores, além das interfaces com os *Shields* que complementam as funcionalidades do microcontrolador (BAYLE, 2013).

Para a correta configuração de leitura de níveis na entrada do Arduino, obrigatoriamente deve-se configurar o pino a ser utilizado. Caso o pino do Arduino seja para leitura de sensores deve-se informar ao código de programação que determinado pino será de entrada, onde a função *pinMode* deve ser atribuída, onde: *pinMode* (10, *INPUT*) quer dizer que o pino 10 está configurado para a leitura e *pinMode* (10, *OUTPUT*) indica que o Arduino irá utilizar o pino 10 para acionamento.

8.1 Níveis lógicos de entrada e saída do Arduino

Estas portas trabalham com sinais digitais e analógicos, ou seja, as digitais reconhecem apenas valores lógicos de 0 à 1, na lógica binária, e este valor segue o padrão *Transistor Transistor Logic* (TTL) onde o range do nível de tensão varia de 0 até a Voltagem em Corrente Contínua (VCC), onde VCC tem a tensão entre 4.75 *Volts* a 5.25 *Volts*, através das funções *DigitalRead* e *DigitalWrite* na programação do Arduino, portas Digitais apelidadas de D0 para a porta 0, D1 para a porta 1 e assim sucessivamente. O nível de tensão entre 0 *Volts* e 0.8 *Volts* as portas digitais do Arduino interpreta como nível 0, convencionado de *LOW* (Nível baixo) ou *OFF* na programação em Arduino. O nível lógico 1 que compreende a tensão de 2 *Volts*, a VCC é conhecida como *High* (Nível alto) ou ON, os valores intermediários são indefinidos e serão reconhecidos para qualquer um dos níveis lógicos (*NATIONAL INSTRUMENTS*, 2016). O Figura 25, representa o gráfico da tensão em nível digital e seus níveis em TTL para o Arduino.

Figura 25 – Gráfico do nível lógico digital



Fonte: Adaptado de (ROCHA, 2014).

Para as análises dos sinais das portas analógicas, os controladores *ATmega* dos Arduinos contam com um conversor integrado analógico digital (A/D) que variam de 6 a 16 canais, dependendo do modelo a ser utilizado. O conversor tem resolução de 10 *bits*, retornando inteiros de 0 a 1023. Desta forma os pinos analógicos, apelidados de A0 para a porta 0, A1 para a porta 1 e assim sucessivamente, podem ser utilizados como pinos Digitais.

Para a programação do Arduino utilizando as portas analógicas, as funções *analogRead*(), *analogWrite*() e *analogReference*() são essenciais.

A função *analogRead*() lê o valor de um pino analógico especificado. Com a resolução de 10 *bits*, ele representa as tensões de entrada do pino, que compreende entre 0 e 5 *Volts*, em informação mapeadas em valores inteiros entre 0 e 1023. Dessa forma, tem-se uma leitura de 5 *Volts* / 1024 = 4,9 m*Volts* por unidade de medida. Essa resolução é devida ao conversor analógico-digital (ADC) utilizado na placa do Arduino que possui uma frequência de 10 KHz, fazendo uma leitura a cada 100 μs (REIS, 2015).

Para um valor mais preciso na leitura dos pinos, deve-se aumentar a resolução. Os valores referentes a faixa de tensão de entrada e a resolução dos pinos pode ser alterada com a função *analogReference*().

A função *analogReference*() é usada na programação do Arduino para ajustar o valor padrão da tensão máxima de referência nas leituras analógicas. Durante a leitura o Arduino faz uma comparação entre a tensão de entrada do pino analógico com um de referência, que por padrão é o valor VCC de alimentação do Arduino, ou seja 5 *Volts* (REIS, 2015).

Esta função pode ser utilizada caso necessite medir valores de tensão em uma faixa distinta, como por exemplo, entre 0 e 3 *Volts*, ou entre 0 e 4,5 *Volts*. Nesse caso, para que o

ADC saiba qual é o valor de tensão máxima da faixa, sendo o inferior o nível zero, usa-se a função *analogReference*().

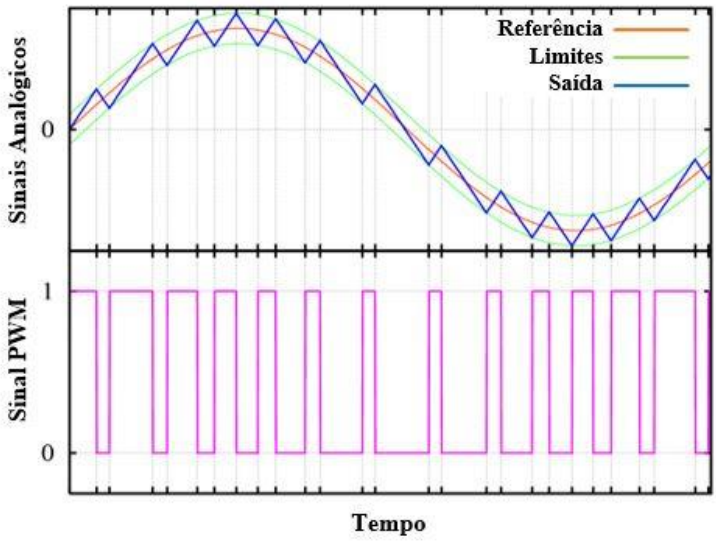
Para o uso da função *analogWrite*() o conceito de PWM precisa abordado, uma vez que os microcontroladores não geram sinais analógicos por si só, eles utilizam esta técnica para gerar os níveis analógicos.

O PWM está presente em todos os microcontroladores modernos pois ao controlar digitalmente os circuitos analógicos reduz-se os custos na implementação do sistema e o consumo de energia. Esta é uma forma de codificar um sinal analógico de forma digital, com esta técnica, através do uso de contadores de alta resolução, o ciclo de trabalho de uma onda quadrada é modulado para codificar um nível de sinal analógico específico para que então ele atenda aos requisitos de uma aplicação desejada, explica Silveira (2013).

Diz-se que o sinal PWM é totalmente digital uma vez que em um determinado instante a alimentação está com o nível alto, ou seja, em 5 *Volts* ou completamente sem tensão, 0 *Volts*. A carga tem a sua fonte de tensão ou corrente fornecida de forma repetitiva onde os impulsos são gerados ora ligados, ora desligados. O tempo ligado é o tempo em que a alimentação é aplicada a carga e o tempo desligado é o período durante o qual a alimentação é desligada. Desta forma qualquer o sinal analógico pode ser codificado com PWM, especificando uma largura de banda suficiente.

Quando se deseja obter um sinal analógico fiel, utiliza-se filtros passa-baixa que são simplesmente um capacitor conectado entre o sinal e o terra, deixando o sinal analógico linear. O gráfico da Figura 26 exemplifica a atuação de um filtro onde a variação PWM de um ciclo de aproximadamente 25% a 75%, temos uma onda próxima de uma onda senoidal. A saída em azul não imita perfeitamente uma onda senoidal mas forma um conjunto de médias locais que atuam como uma onda senoidal (SILVEIRA, 2013).

Figura 26 – Aplicação de filtro para o PWM



Fonte: (SILVEIRA, 2013).

Desta forma, a função *analogWrite*() trabalha o sinal na saída de um pino Analógico específico enviando um sinal PWM que permite escrever um nível analógico em um pino. Geralmente esta função é utilizada para ativar e ou desativar dispositivos conectados ao Arduino.

A função *analogWrite*() gera um pulso de sinal de onda quadrada em uma razão cíclica (*duty cicle*) determinada, até que uma nova chamada à função seja realizada. A frequência do sinal PWM na maioria dos pinos é de cerca de 490 Hz. A Tabela 01 mostra os principais microcontroladores, a frequência de PWM e os pinos utilizados.

Tabela 01 – Frequência e pinos PWM

Arduino	Controlador	Frequência (Hz)	Pinos
UNO	ATMega328	980	5 e 6
LEONARDO	ATMega32u4	980	3 e 11
MEGA	ATMega1280	980	de 2 à 13 e de 44 à 46

Fonte: (SILVEIRA, 2013).

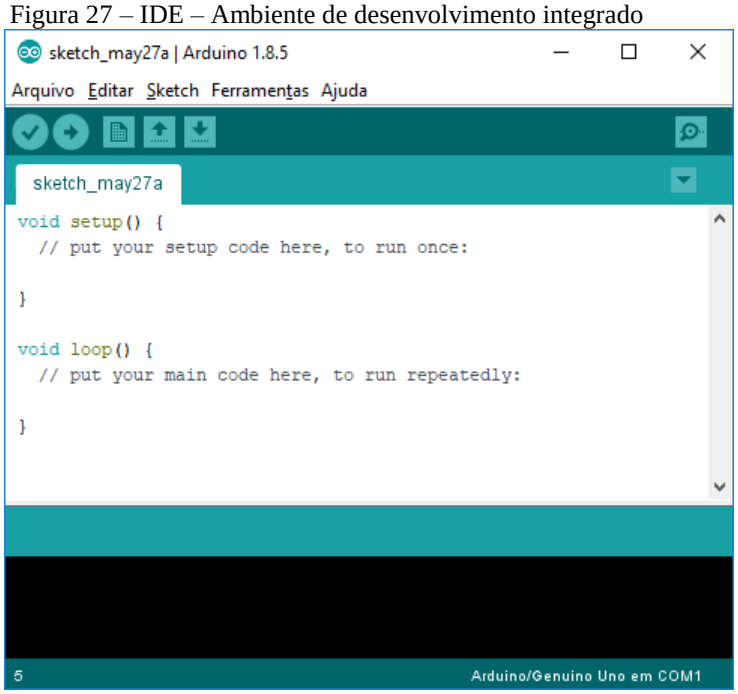
Uma forma de identificar os pinos correspondentes ao PWM diretamente na placa, é só visualizar o sinal ~ ao lado do número do pino.

8.2 Ambiente de desenvolvimento - IDE

O microcontrolador Arduino é um dispositivo formado principalmente por 2 componentes básicos: a placa de circuito, que é o elemento de *hardware* utilizado para construir

as configurações físicas e interligação de módulos e sensores e o outro elemento de grande importância e ao IDE, conforme a Figura 27, que é um *software* desenvolvido em conjunto com a comunidade Arduino para escrever os códigos fontes que interpretam os resultados obtidos pelos módulos e sensores através das portas de entrada e representam os resultados nas portas de saída conforme a lógica de programação. Estes códigos podem ser escritos nas linguagem de programação C/C++ e salvos com extensão de arquivo INO (ARDUINO.CC, 2017).

Figura 27 – IDE – Ambiente de desenvolvimento integrado



Fonte: (BAYLE, 2013, p. 30).

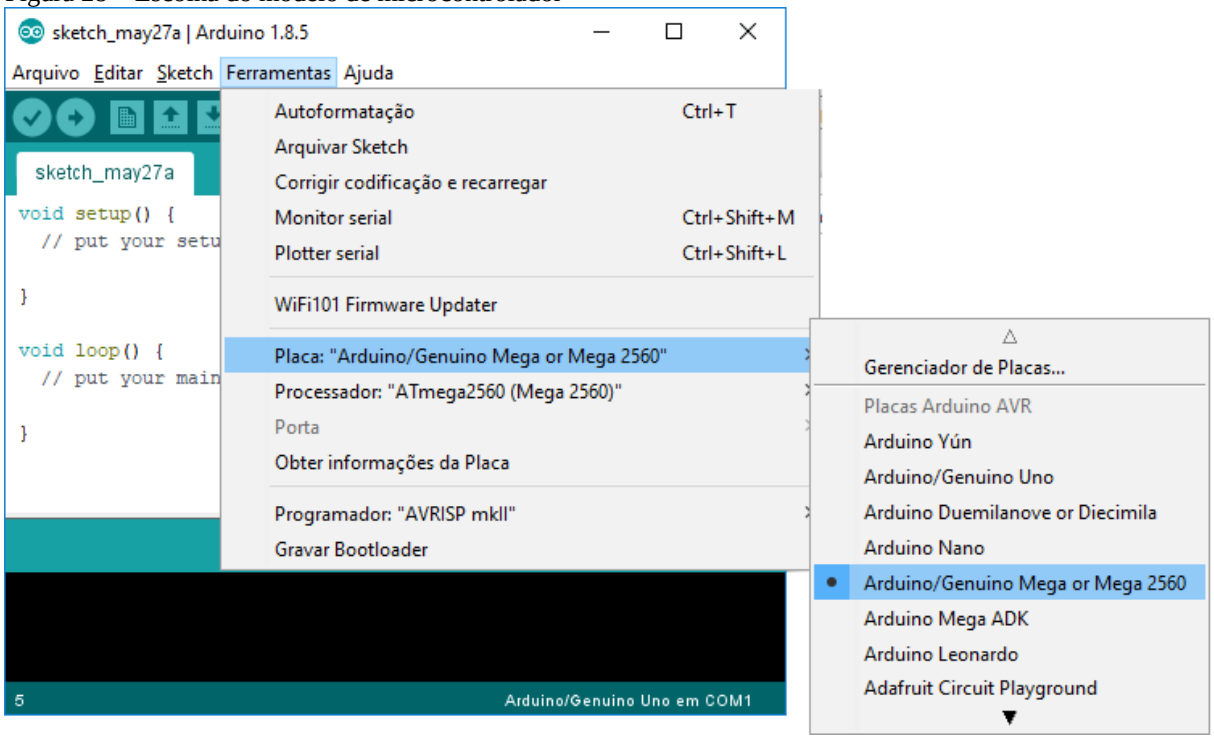
Este ambiente de desenvolvimento é disponibilizado de forma *Free* e pode ser adquirido no site do fabricante para diversas versões de sistemas operacionais com *Windows* em 32 e 64 *bits*, *Linux*, também de 32 e 64 *bits* e *Mac OS* 10.7 *Lion* ou mais recente. Outra versão disponível é *online*, onde o programador tem a opção de escrever a sua aplicação diretamente "nas nuvens" bastando para isso o cadastro de um usuário e senha (ARDUINO.CC, 2017).

Programas escritos no IDE do Arduino são chamados *Sketches*, que ao serem compilados para uma linguagem que o Arduino reconhece, o *software* faz a transferência para o microcontrolador que deve estar conectado a uma porta USB do computador. Através do IDE.

Para o IDE transferir o *Sketch* de forma correta, antes de fazer o *upload*, deve-se identificar no *software* qual a placa de microcontrolador está conectada. Assim o ambiente de desenvolvimento controla o que está sendo transferido com o tamanho de memória do *hardware* que irá receber o código.

Para a definição de qual modelo está sendo desenvolvido, o IDE fornece todos os microprocessadores disponíveis para programação, conforme a Figura 28.

Figura 28 – Escolha do modelo de microcontrolador

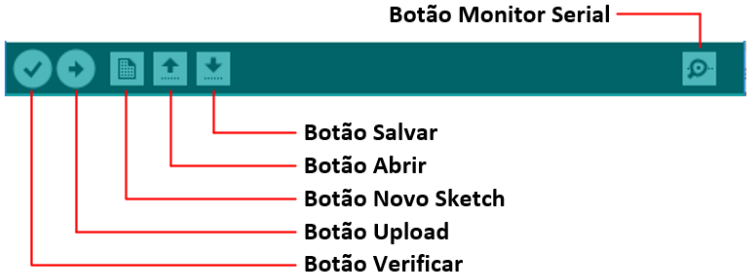


Fonte: (BAYLE, 2013, p. 30).

O ambiente de desenvolvimento é dividido em três partes: a barra de ferramentas no topo, a área de criação e edição do código ou a janela *Sketch* no centro, e a janela de mensagens na parte inferior. Ele fornece uma lista de *Sketches* de exemplo que auxilia o desenvolvedor para programar em diversos periféricos, fornecendo uma base uma base a partir da qual poderão construir seus próprios códigos (MCROBERTS, 2011).

A barra de ferramentas consiste de cinco botões, Verificar, *Upload*, Novo *Sketch*, Abrir, Salvar e *Monitor Serial*, conforme a Figura 29. Estes botões fornecem acesso as principais funções do IDE.

Figura 29 – Barra de ferramentas



Fonte: O autor.

Depois do código escrito no IDE, usa-se o botão Verificar para verificar se existe algum erro no código antes de fazer o *upload* para a placa do Arduino.

O botão Novo cria um novo *Sketch* em branco, pronto para o programador iniciar um novo código. O IDE solicita um nome e localização para o *Sketch*.

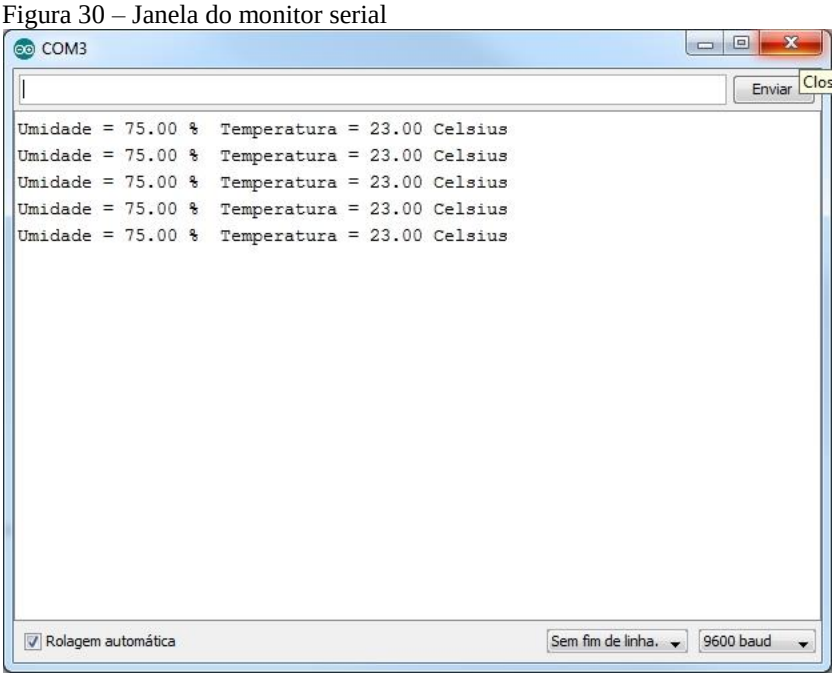
O botão *Open* abre uma janela para o desenvolvedor selecionar o arquivos previamente salvo para a edição.

O botão *Save* salva o código editado no arquivo, informando *Done Saving* (Gravação Completa) ao concluir o comando.

O botão *Upload* carrega o código desenvolvido para o Arduino, porém antes o IDE executa algumas rotinas para certificar da integridade dos dados compilados. Ele verifica se existe algum erro, compila o código e faz o *upload*.

O botão *Monitor Serial* fornece uma ferramenta muito útil. Ela abre uma janela que exibe os dados seriais enviados do Arduino, seja este conectado pela USB ou placa serial. A principal função desta ferramenta é poder depurar o código e ter a possibilidade de enviar comandos para o Arduino. A Figura 30 mostra a janela de monitor serial e para o funcionamento correto deve-se ajustar a taxa de transmissão (*Baud Rate*). Esta valor de transmissão é a taxa por segundo de comunicação entre o Arduino e o computador onde ele está conectado. Por padrão este valor é definido em 9600 *Baud* e segundo McRoberts (2011), se for enviado determinado texto pela linha de comunicação serial, cabo USB ou Serial, 1200 letras ou símbolos de texto serão enviados por segundo pois 9600 *bits* / 8 *bits* = 1200 *Bytes* ou caracteres.

Figura 30 – Janela do monitor serial



Fonte: (ARDUINO E CIA, 2016).

O IDE do microcontrolador Arduino não tem segredo para a sua utilização, é limpo, claro e fácil.

8.3 Programação do Arduino

Os códigos de programação para o ambiente Arduino são desenvolvidos utilizando a linguagem C, de acordo com Bayle (2013), um programa é o texto que se escreve usando uma determinada linguagem de programação que contém comportamentos apropriados para um processador, é criar uma maneira de lidar com valores fornecidos, de entrada, e reproduzi-los na saída de acordo com este comportamento.

A linguagem C é estruturada e de baixo nível, ou seja, tem acesso a instruções diretas aos hardwares dos processadores onde se destina o código de programação.

A programação para Arduino é feita usando as sintaxes e expressões baseadas na linguagem C e quase todas as bibliotecas do Arduino, que são funções e classes reutilizáveis, são desenvolvidas em C++ que pode-se dizer que é uma linguagem C com orientação a objetos, onde cria-se as classes que são modelos de funções e chama-se estas classes dentro do código criando as instancias dessa classe que tem vida própria dentro do tempo de execução e respeitam a herdam a estrutura da classe de onde se originaram (BAYLE, 2013).

Para que o Arduino armazene informações e valores durante a sua execução, faz uso das *Variáveis*, que é um local de armazenamento de memória onde um nome é atribuído para representa-la. Fica então reservado uma área da memória que pode ser preenchida ou deixada vazia. Basicamente, é usado para armazenar diferentes tipos de valores, podendo alterar o seu conteúdo (o valor) em tempo de execução. Em contra partida existem as *Constantes* que tem as mesmas características das variáveis, ocupam lugar na memória, mas que não pode-se alterar o seu valor durante a execução do programa (BAYLE, 2013).

Variáveis e *Constantes* estão associadas a um tipo, também chamado de tipo de dados, que define a natureza dos dados e também reserva diretamente um espaço com um tamanho definido na memória. A linguagem C tem cerca de 10 tipos principais de dados conforme descrito na Tabela 02, que são muito utilizados na programação do Arduino.

Tabela 02 – Tipos de variáveis e constantes

Tipo	Definição	Tamanho na memória
`boolean`	Armazena valor `True` (Verdadeiro) ou `False` (Falso).	1 Byte (8 bits)
`char`	Armazena caracteres com aspas simples, como 'a' ou número, seguindo a tabela ASCII. É um tipo sinalizado e armazena números de -128 a 127, ou pode ser não sinalizado e armazenar números de 0 a 255.	1 Byte (8 bits)
`byte`	Armazena números de 8-bits não sinalizados de 0 a 255	8 bits
`int`	Armazena números com 2 Bytes sinalizados que variam de -32.768 a 32.767 e que podem ser não sinalizados variando de 0 a 65.535.	2 Bytes (16 bits)
`long`	Armazena números com 4 Bytes sinalizados que variam de -2.147.483.648 até 2.147.483.647 a podem ser não sinalizados e seu valor variar de 0 a 4.294.967.295	4 Bytes (32 bits)
`float`	Basicamente armazena números com um ponto decimal de -3.4028235E + 38 a 3.4028235E + 38 como 4 Bytes sinalizados.	4 Bytes (32 bits)
`double`	Geralmente armazena valores flutuantes com precisão de duas vezes maior que o valor `float`.	4 Bytes (32 bits)
`Array`	`Array` é uma estrutura ordenada de consecutivos elementos do mesmo tipo que pode ser acessado com um número de index	Nº elementos x tamanho do tipo do elemento
`string`	Ele armazena palavras em uma matriz de char onde o último elemento é nulo que é um caractere particular (Código ASCII 0).	Nº elementos X 1 Byte

Fonte: (BAYLE, 2013, p. 212)

A linguagem C/C++ permite uma ampla variedade de estruturas de controle de fluxo de processamento. Os “Operadores” indicam ao compilador a necessidade de se fazer cálculos matemáticas ou lógicas para se definir qual o decisão a se tomar dependendo do valor recebido.

A Tabela 03 exprime os tipos de “Operadores” utilizados.

Tabela 03 – Operadores

Operadores

Tipo	Operador	Propósito	Exemplo
Aritméticos	`+`	Adição	`a = 4 + 1; // 5`
	`-`	Subtração	`a = 4 - 1; // 3`
	`*`	Multiplicação	`a = 2 * 4; // 8`
	`/`	Divisão	`a = 8 / 2; // 4`
	`%`	Módulo (resto da divisão)	`a = 5 % 2; // 1`
Atribuição	`=`	Atribuição simples	`a = 50;`
Lógicos	`&&`	“e” lógico	`(a > 1) && (b < 1)`
	`||`	“ou” lógico	`(a > 1) || (b < 1)`
	`!`	não (inversão)	`!(a > 2)`
Relacionais (Comparação)	`==`	igual a	`(a == 0)`
	`!=`	diferente de	`(a != 0)`
	`<`	menor que	`(a < 0)`
	`>`	maior que	`(a > 0)`
	`<=`	menor ou igual a	`(a <= 0)`
	`>=`	maior ou igual a	`(a >= 0)`
Incremento e Decremento	`++`	Incremento	`a++;`
	`--`	Decremento	`a--;`

Referência (Apontadores) Operadores utilizados antes do nome de variáveis	`&`	Retorna o “endereço de”	`int a; //variável inteira`
			`int *p; //declaração de ponteiro`
			`p = &a; //atribui o endereço de a`
			`*p = 2; //atribui ao conteúdo`
			`//apontado por p o valor 2;`
			`//como p aponta para o endereço`
			`//de a, então a recebe 2.`

	`*`	Retorna o “conteúdo de”	

Fonte: (BAYLE, 2013, p. 287).

Além dos “Operadores”, a programação oferecida pelo IDE do microcontrolador Arduino também consta com as estruturas de controle onde é possível definir qual bloco de instruções será executado e quantas vezes ele irá rodar. E o que denomina condição de controle, onde é possível através de uma expressão lógica ou aritmética, avaliar se o resultado retornará verdadeiro ou falso. A Tabela 04 detalha as expressões de controle.

Tabela 04 – Condição de controle

Comandos da Linguagem

Comando	Propósito	Sintaxe

if	Comando condicional	`if (a > b) {`
		`  printf("%s", "A é maior que B");`
		`} else {`
		`  printf("%s", "A é igual ou menor que B");`
		`}`

switch	Comando condicional	`switch (i) {`
		`  case 0 :`
		`  printf("%s", "ZERO");`
		`  break;`
		`  case 1:`
		`  printf("%s", "UM");`
		`  break;`
		`  case 2:`
		`  printf("%s", "DOIS");`
		`  break;`
		`}`

while	Laço com pré validação	`int i = 1;`
		`while (i <= 10) {`
		`  printf("%d", i++);`
		`}`

do	Laço com pós validação	`int i = 1;`
		`do {`
		`  printf("%d", i++);`
		`} while (i <= 10);`

for	Laço simplificado	`for (i=1;i<=10;i++){`
		`  printf("\n%d", i);`
		`}`

break	Saída de bloco	`break;`
continue	Reinício de bloco	`continue;`

Fonte: (BAYLE, 2013, p. 308).

Para que o Arduino realize as leituras e efetua alguma ação nos pinos, seja digital ou analógico, é preciso que algumas funções sejam declaradas e que aponte o pino e o valor a ser lido ou atribuído. Na estrutura do microcontrolador Arduino existem várias destas funções prontas para o uso e não necessita de muita programação para utiliza-las.

8.4 Módulos, sensores e *Shields* para o Arduino

Apenas com a placa de microcontrolador Arduino o programador fica limitado a desenvolver suas aplicações utilizando apenas os recursos disponíveis no controlador, mas para

ampliar as suas funcionalidades, estão disponíveis módulos, sensores e *Shields* que trabalham como interfaces para expandir as possibilidades do Arduino.

Os módulos mais conhecidos são para comunicação via tecnologia *Bluetooth*, módulos de relés, *display* de LCD, *joystick*, detector de chuva e até mesmo câmeras de captação de vídeo.

Os sensores disponíveis são para medir distância, movimento, obstáculo, sensor de velocidade, umidade do ar, umidade do solo, sensor de chama, etc.

Com relação aos *Shields* que se acoplam a placa Arduino os mais comuns são os *Ethernet Shield* para comunicação em rede, *Thin Film Transistor* (TFT), *TouchScreen Display Shield*, *Arduino Motor Shield*, 3G GPS comunicação *Shield*, Motor *Shield*, etc.

Para um desenvolvimento de sensoriamento de solo na agricultura, baseado em sua umidade, os principais módulos, sensores e *Shield* são necessários conforme os detalhamentos.

8.4.1 Módulo de *display* LCD

O *Liquid Crystal Display* (LCD), conforme a Figura 31 é um módulo de grande utilização na maioria dos projetos desenvolvidos em Arduino. Através deste componente é possível fazer a interface homem máquina e exibir informações relevantes ao operador. Ele possui 16 pinos, conforme a Tabela 05, por onde o *display* recebe a alimentação elétrica e faz a transferência dos dados para a exibição.

Figura 31 – Módulo de *display*

Fonte: Adaptado de (FRITZING, 2018).

Tabela 05 – Pinagem do *display* LCD

Pino	Sigla	Descrição
1	GND	Terra (0V)
2	VCC	5 Volts
3	V0	Ajuste de Contraste
4	RS	Seleção de registro
5	R/W	Leitura e Escrita
6	E	Enable
7	D0	Pinos de dados
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	
15	A	Anodo
16	K	Catodo

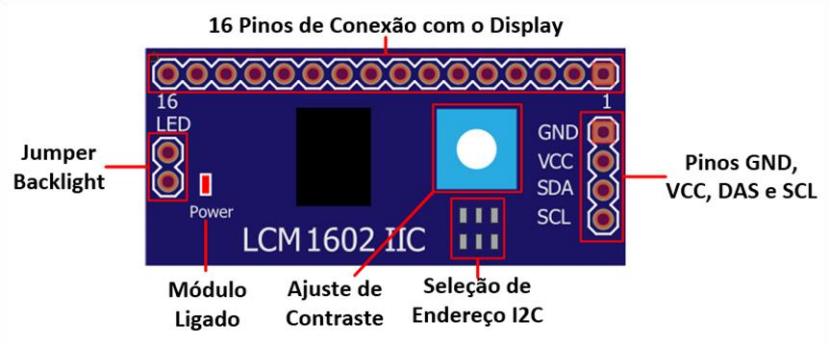
Fonte: (ARDUINO E CIA, 2016).

Para facilitar a programação do Arduino com o *display* LCD, um outro módulo, denominado Serial I2C, diminui a quantidade de pinos utilizados com o Arduino.

8.4.2 Módulo I2C serial LCD *display* (PCF8574)

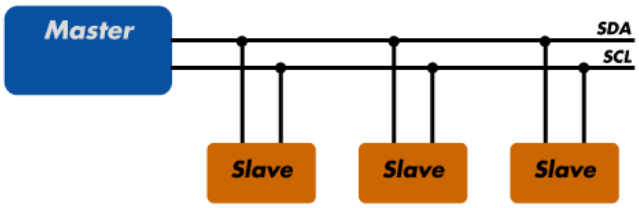
Através deste módulo, conforme a Figura 32, é possível controlar um *display* de 16 colunas por 2 linhas ou de 20 colunas por 4 linhas utilizando apenas dois pinos do Arduino, o pino analógico 4 *Serial Data* (SDA) e o pino analógico 5 *Serial Clock* (SCL), que juntos formam a interface de comunicação I2C, que segundo Reis (2015), trabalha no regime *Master/Slave* em barramento, conforme Figura 33 e pode comunicar-se com até 112 dispositivos, dentre eles o *Display* de LCD.

Figura 32 – Módulo I2C serial LCD *display*



Fonte: Adaptado de (ARDUINO E CIA, 2016).

Figura 33 – Barramento I2C



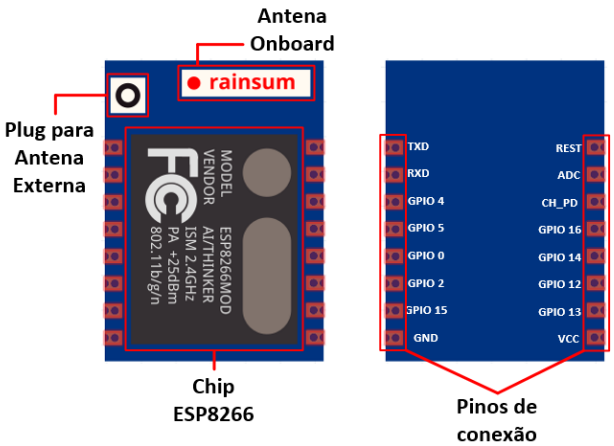
Fonte: (REIS, 2015).

A forma como se trabalha com o I2C é através de endereçamento, onde cada dispositivo *Slave* tem um valor atribuído que o identifica e o torna único no sistema de comunicação. Desta forma é possível com uma única placa Arduino conectar-se a diversos dispositivos como *displays* LCD, sensores de temperatura, acelerômetros, *Real Time Clock* (RTC), conversores *Analog Digital Converter* (ADC) e *Digital Analog Converter* (DAC), simultaneamente onde cada dispositivo tem o seu Identificador (ID) que é único no mesmo barramento.

8.4.3 Módulo de comunicação WIFI ESP8266

O módulo wireless ESP8266, de acordo com a Figura 34, permite conectar o microcontrolador Arduino às redes WIFI no padrão 802.11 b/g/n, que são as redes mais comuns utilizadas para acessar a *Internet*. Este módulo trabalha como *Access Point* (AP) e também como *Station* (STA), no envio e recepção de dados.

Figura 34 – Módulo WIFI ESP8266



Fonte: Adaptado de (MINATEL, 2017).

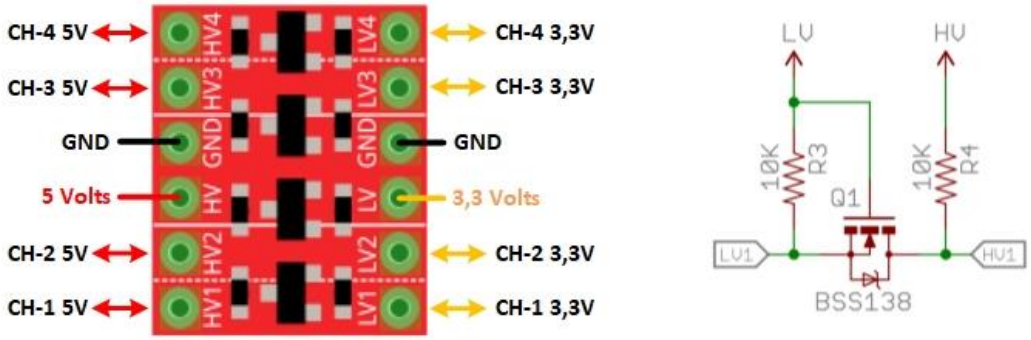
As principais características de um módulo WIFI são:

a) Suporte à redes nos padrões 802.11 B/G/N;

b) Alcance de comunicação aproximada: 85 metros, em ambiente aberto;
c) Tensão: 3.3 VDC;
d) Comunicação serial pelos pinos TX e RX;
e) Modos de operação: Cliente, *Access Point*, Cliente + *Access Point*;
f) Modos de segurança *wireless* : *OPEN* / *WEP* / *WPA_PSK* / *WPA2_PSK* / *WPA_WPA2_PSK*;
g) Operando nos modos TCP/UDP, permite até 5 conexões simultâneas;

O ESP8266 foi projetado para trabalhar com níveis de tensão de 3,3 *Volts* no recepção dos dados de comunicação, o que leva ao projetista uma atenção ao usá-lo com a maioria dos modelos de microcontroladores Arduino que trabalham com nível de 5 *Volts*. Para tornar esta comunicação possível utiliza-se um conversor de nível lógico, bidirecional de quatro canais disponível para este tipo de aplicação, conforme a Figura 35.

Figura 35 – Conversor bidirecional de nível lógico de quatro canais



Fonte: (ALBERTS, 2013. Pág. 01)

Este *buffer* que ao ser alimentado pela tensão de 3,3 *Volts*, todos as suas 6 portas lógicas inversoras utilizaram esta tensão como referência, e ao utilizar a tensão de 5 *Volts* do Arduino na entrada de cada porta ele representa em 3,3 *Volts* na sua saída.

8.4.4 Transceptor com comunicação via rádio

Para uma melhor organização na infraestrutura de sensores, principalmente aqueles que são disponibilizados para operarem de forma à monitorar uma grande área, locais onde haja riscos e de difícil acesso ou onde a quantidade de sensores torna a sua ligação à fio inviável, faz-se o uso da tecnologia de Redes de Sensores Sem Fio (RSSF).

O transceptor nRF24L01+ da fabricante norueguesa *Nordic Semicondutores*, conforme a Figura 36, que tem seu foco de atuação em dispositivos sem fio de baixo consumo, é um dispositivo que leva o nome de transceptor por ter a característica de utilizar o mesmo

equipamento para transmissão e recepção de dados, projetado para operação na faixa de frequência *Industrial Scientific and Medical* (ISM) que segundo Teleco (2018), são faixas de frequência reservadas internacionalmente para o desenvolvimento Industrial, científico e médico.

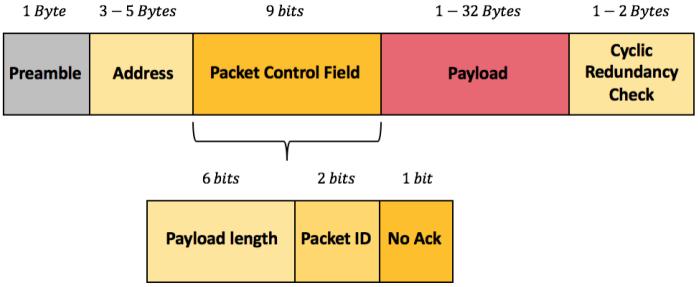
As características de construção de hardware do transceptor, constitui de módulos com a possibilidade de ligação de antena externa, o que aumenta consideravelmente o seu alcance, chegando à 1000 metros de distância, sem obstáculos.

Figura 36 – Rádio transceptor modelo nRF24L01+ com antena externa

Fonte: Adaptado de (DEJAN, 2017).

Nordic Semicondutores (2018), afirma que este modelo de rádio pode ser utilizado com qualquer dispositivo que possua uma ICSP, como no caso dos microcontroladores Arduino, podendo suportar taxas de transmissão de dados de 250 Kbps, 1 Mbps, podendo chegar aos 2 Mbps. Como este transceptor pode trabalhar em uma estrutura *Master*/*Slave* (Mestre / Escravo), onde um receptor pode receber informações de até 6 transmissores, existe o registro de mapeamento, que é acessível através da comunicação ISP, e que contém todos os registros de configuração no nRF24L01+. Sua estrutura de pacote de divide em 5 campos diferentes, conforme a Figura 37 (NORDIC SEMICONDUTORES, 2018).

Figura 37 – Estrutura do pacote de comunicação do nRF24L1+



Fonte: (FRASER, 2017).

Essa estrutura permite uma carga de transmissão de dados, denominada *Payloads* de tamanho variável, que pode variar 1 à 32 Bytes. Este pacote carrega toda informação útil da comunicação e por ela o transceptor faz a distinção entre os componentes do sistema e todo o controle de no caso existir vários elementos comunicando simultaneamente. Isto significa que um sistema de comunicações bidirecionais pode ser estabelecido inteiramente usando um dispositivo *Master* que transmite solicitações de dados para cada dispositivo *Slave* reconhecendo cada membro desta comunicação. (NORDIC SEMICONDUTORES, 2018).

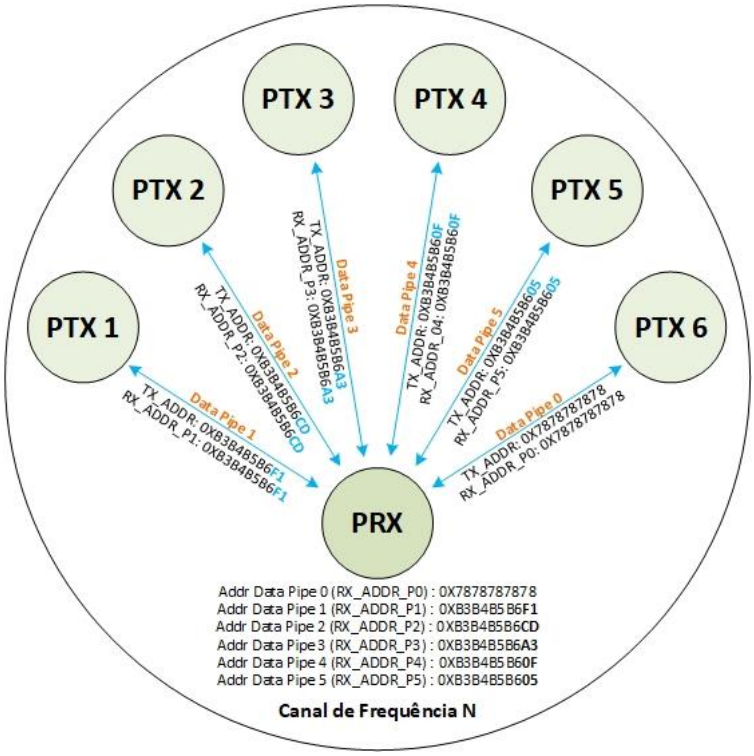
Para que o sistema funcione com múltiplos dispositivos transmitindo para um receptor, o recurso de *MultiCeiver* ™ é usado. Desta forma são formados os *Cluster* e cada componente de arranjo é denominado de *Pipe*.

O receptor contém um conjunto de seis canais de dados paralelos com endereços exclusivos, o que permite 6 conexões simultâneas, conforme Figura 38.

Um *Pipe* de dados é um canal lógico no canal Rádio Frequência (RF) físico, onde cada canal de dados tem seu próprio endereço físico (endereço do *Pipe* de dados).

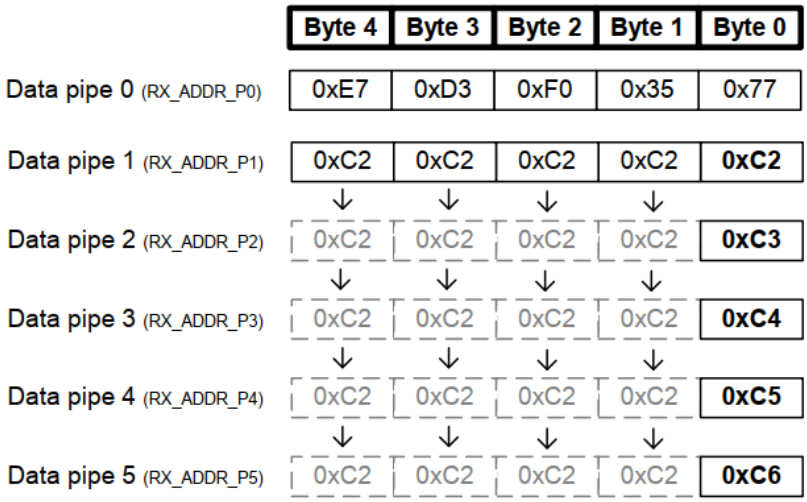
Cada *Pipe* pode ter até um endereço configurável de 5 *Bytes*. O *Pipe* de dados 0 tem um endereço único de 5 *Bytes*, já os *Pipes* de 1-5, estes compartilham os quatro *Bytes* de endereço mais significativos. O LSByte deve ser exclusivo para todos os seis *Pipes*. A Figura 39, é um exemplo de como os *Pipes* de dados são endereçados (NORDIC SEMICONDUTORES, 2018).

Figura 38 – Conexões com múltiplos transmissores



Fonte: Adaptado de (FRASER, 2017).

Figura 39 – Endereçamento dos *Pipes*



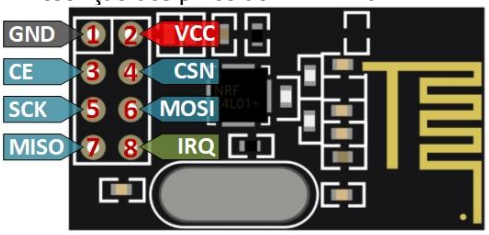
Fonte: (FRASER, 2017).

O PRX recebe os pacotes de mais de um PTX. Para garantir que o pacote do PRX seja transmitido para o PTX corretamente, ou seja, entregue a resposta ao PRX que o solicitou, ele usa o canal de dados endereço onde recebeu o pacote e usa-o como o endereço TX ao transmitir o pacote. No PRX o endereço RX_ADDR_Pn, definido é o endereço do canal e deve ser exclusivo. No PTX, o TX_ADDR deve ser o mesmo que o RX_ADDR_Pn e como o endereço do *Pipe* para o *Pipe* designado.

O nRF24L01+ possui algumas características importantes que torna a sua utilização confiável, uma delas é poder confirmar o recebimento dos pacotes com as informações trocadas entre os Nodos e ser configurado para fazer o reenvio deste pacote caso haja perda, podendo a chegar a 15 reenvio.

Para a ligação de comunicação com o microcontrolador Arduino, o transceptor consiste dos seguintes pinos ilustrado pela figura 40:

Figura 40 – Descrição dos pinos do nRF24L01+



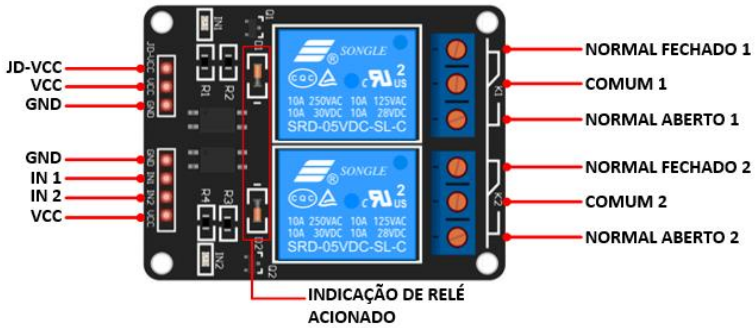
PINO	NOME	FUNÇÃO DO PINO	DESCRIÇÃO
1	GND	Alimentação	Ground (0V)
2	VDD	Alimentação	Fonte de Alimentação (+1.9V - 3,6V DC)
3	CE	Entrada Digital	Modo que ativa RX ou TX para o Chip
4	CSN	Entrada Digital	Seleção SPI para o Chip
5	SCK	Entrada Digital	SPI Clock
6	MOSI	Entrada Digital	Entrada de dados SPI Escravo
7	MISO	Saída Digital	Saída de dados SPI Escravo, com opção Tri-State
8	IRQ	Saída Digital	Pino de interrupção mascarável com ativação em nível baixo

Fonte: Adaptado de (NORDIC SEMICONDUTORES, 2018).

8.4.5 Módulo de relé

Os relés, como pode ser visto na Figura 41, são chaves/interruptores que abrem e fecham contatos eletromecânicos ou eletrônicos. O microcontrolador Arduino por não suportar correntes altas nas suas pinagens, para acionamento de motores por exemplo, faz-se uso deste dispositivo para tal função. Quando um contato de relé está aberto, também chamado de normalmente aberto ou *Normally Open* (NO), há um contato aberto quando o relé não está energizado. Quando um contato de relé está fechado, também conhecido como normalmente fechado ou *Normally Close* (NC), há um contato fechado quando o relé não está energizado. Em ambos os casos, a aplicação de corrente elétrica passando pela bobina do relé altera este seu estado.

Figura 41 – Módulo de relé

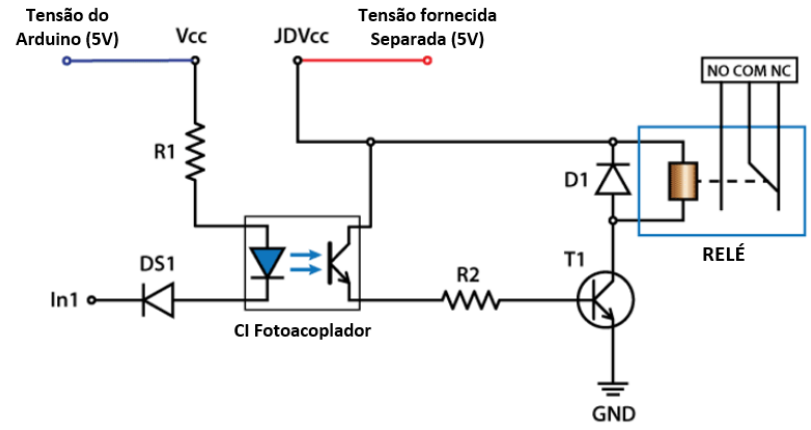


Fonte: Adaptado de (FERNANDO K TECNOLOGIA, 2015).

Geralmente, os relés são usados para alternar correntes menores em um circuito de controle e geralmente não controlam dispositivos que consomem energia, exceto para motores pequenos, sirenes, alarmes luminosos e solenoides que consomem baixas correntes. Estes relés suportam cargas de até 10 A para 250 em Voltagem de Corrente Alternada (VAC), 125 VAC ou 30 VDC. Outro ponto relevante é que para cada relé no módulo existe um circuito de proteção para evitar avarias ao Arduino.

A Figura 42 exemplifica a estrutura para acionamento de um módulo de relé eletromecânico.

Figura 42 – Estrutura de acionamento do módulo de relé



Fonte: (NEDELKOVSKI, 2017).

8.4.6 Dispositivos de comando – Contatores

Para cada tipo de plantio, utiliza-se um determinado tipo de equipamento para acionamento do sistema de irrigação que podem ser vários, destaca-se os principais, o volume de água a ser utilizado, a distância entre a instalação dos aspersores ou até mesmo a dimensão da área atendida pela irrigação. Em muitos casos a utilização de equipamentos de grande potência, como as motobombas, que geralmente são motores de alimentação elétrica trifásica e que requerem dispositivos para o seu acionamento.

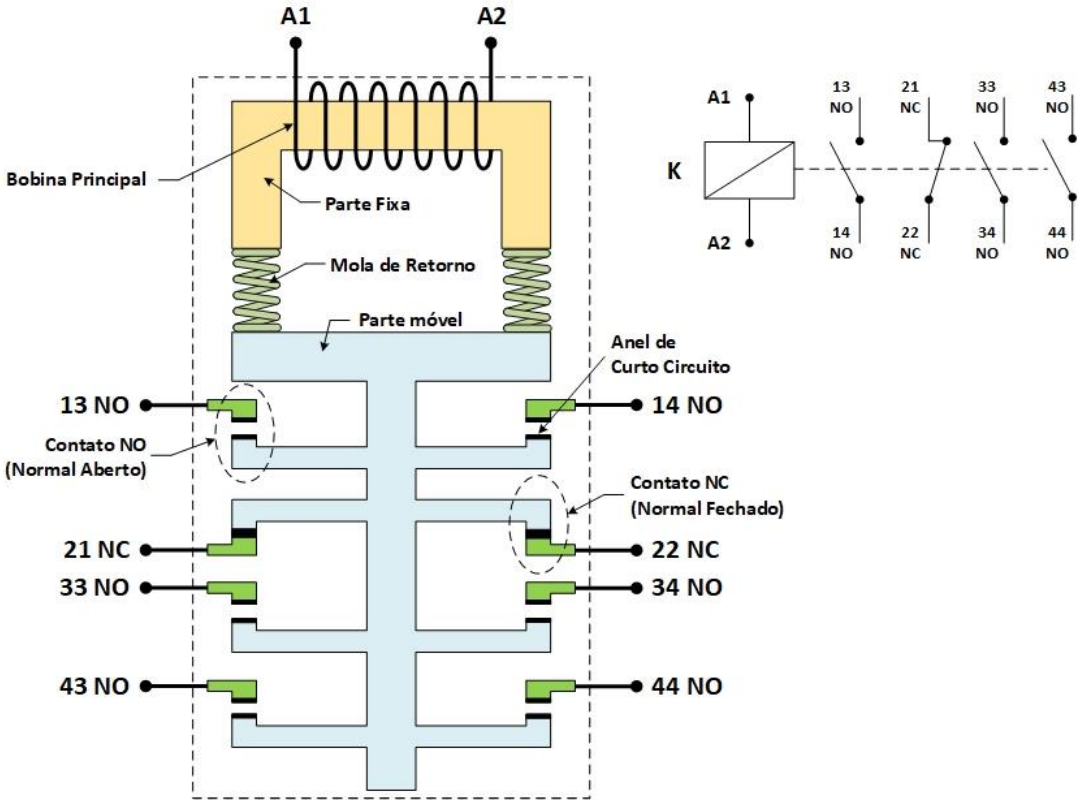
Segundo Athos Eletronics (2015), o Contator é um dispositivo eletromecânico, que é usado para controlar cargas em um circuito de potência à partir de um circuito de comando. Os Contatores permitem, por exemplo, realizar partida direta em motores trifásicos, e permitem ser arranjados de para constituir uma partida indireta. Apesar de amplamente utilizado para o acionamento de motores elétricos, o Contator pode ser usado em diversas outras aplicações, conforme a Figura 43.

Figura 43 – Contator WEG modelo CAW04 31E

Fonte: (WEG, 2017).

Estes dispositivos são montados com um conjunto de contatos fixos denominado A, e um outro de contatos móveis denominado B, que se completam para formar o circuito de acionamento abrindo e fechando estes contatos. A parte móvel é controlada por um núcleo de ferro, envolvido por uma bobina que ao ser energizada cria um campo magnético que movimenta a parte móvel realizando os contatos para abrir ou fechar contatos. Usualmente os Contatores possuem um conjunto de contatos normalmente fechados NC que “abrem” quando a bobina é energizada e um conjunto de contatos normalmente abertos NO que "fecham" na mesma situação, retornando ao estado “normal” quando a bobina é desenergizada, conforme diagrama esquemático da Figura 44.

Figura 44 – Esquema elétrico do Contator



Fonte: Adaptado de (SEAAN, 2014).

Para determinar qual Contator é o ideal para uma aplicação, os seguintes quesitos são observados como, a tensão de aplicação da bobina que pode ser de 24 VCC, 127 VCA, 220 VCA, 380 VCA, 440 VCA, os números de contatos e sua característica “Normalmente Aberto” ou “Normalmente fechado”, tensão de isolação e a corrente nominal de emprego, ou seja, a corrente do Contator. Outro item fundamental para o dimensionamento, reforça Souza (2014),

é muito importante observar a categoria de aplicação para o qual o Contator está sendo especificado, conforme a Tabela 06.

Tabela 06 – Categoria dos Contatores

CATEGORIA	APLICAÇÃO
AC1	Manobras leves, cargas pouco indutivas (Aquecedores, Lâmpadas incandescentes, Lâmpadas fluorescentes, etc).
AC2	Manobras leves, manobras de motores (Bombas, Guinchos e Compressores), desligamento em regime.
AC3	Serviço normal de manobras de motores com rotor de gaiolas (Bombas, Ventiladores, Compressores), desligamento em regime.
AC4	Manobras pesadas, acionamento de motores em plena carga (Pontes rolantes, Tornos, etc), comando intermitente (Pulsatório), Reversão em plena carga.

Fonte: (SOUZA, 2014).

Para determinar a corrente de emprego do Contator, faz-se o cálculo, conforme a equação 1.

$$le \geq ln\ x\ 1{,}15 \qquad (01)$$

Onde:

le: Corrente nominal de emprego (Corrente do Contator);

ln: Corrente nominal do motor;

1,15: Fator de segurança. Acréscimo de 15% da corrente de trabalho do Contator.

8.4.7 Válvula solenoide

A válvula solenoide é um sistema eletromecânico controlado, conforme a Figura 45. Sua principal característica solenoide se deve a uma bobina elétrica com um núcleo ferromagnético móvel no centro, sendo este núcleo denominado êmbolo. Em uma posição de repouso, o êmbolo tampa um pequeno orifício por onde é capaz de circular um fluído, que no caso deste projeto este fluído é a água. Quando uma corrente elétrica circula através da bobina, um campo magnético é criado que por sua vez exerce uma força no êmbolo. Com isso, este dispositivo é puxado em direção ao centro da bobina de modo que o orifício se abre deixando passar a água.

Figura 45 – Módulo de válvula solenoide

Fonte: (FILIPEFLOP, 2017).

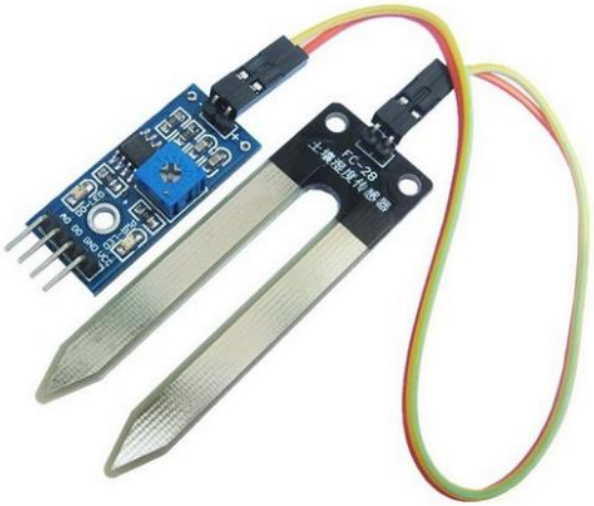
Principais Especificações:

a) Tensão de Operação: 12 VDC;
b) Corrente Nominal: 500 mAh;
c) Pressão de Operação: 0,2 a 6 kgf/cm^2;
d) Vazão Mínima= 7 litros/min (a 0,2kgf/cm^2);
e) Vazão Máxima= 40 litros/min (a 6 kgf/cm^2);
f) Temperatura Máxima do Fluído: 60º Celsius;
g) Entrada: rosca externa de 3/4" (25mm);
h) Saída: rosca externa de 3/4" (25mm);

8.4.8 Sensor de umidade do solo higrômetro

O Sensor de Umidade do Solo Higrômetro, conforme a Figura 46, é um dos sensores mais simples e mais baratos oferecidos no mercado para monitoração do solo. Alimentando apenas o dispositivo com uma tensão VCC que pode variar de 3,3 a 5 *Volts* e GND, o sensor está pronto para fazer as amostras e retornar pelo pino D0 (Digital) ou A0 (Analógico), por onde o Arduino irá obter o valor lido, este valor de retorno do sensor varia dependendo da tensão de alimentação do sensor.

Figura 46 – Sensor de umidade do solo higrômetro



Fonte: (ARDUINOLANDIA, 2018).

As duas pequenas hastes atuam como um resistor variável, quanto mais úmido o solo melhor condutividade, o que resulta em menor resistência e maior valor de sinal.

Este modelo de sensor oferece duas formas de entregar a leitura ao Arduino, uma Digital pelo pino D0, onde através do módulo comparador, modelo LM393 que converte os valores analógicos para digital, quando o solo está seco a saída permanece em estado ALTO, valor um, ou seja VCC. Caso contrário se o solo estiver úmido, a saída fica em estado baixo, valor 0. O limite entre estas extremidades pode ser ajustado através de um potenciômetro no módulo que regulará a saída Digital. Outra maneira de fazer a leitura é totalmente analógica pelo pino A0, que oferece uma melhor resolução e precisão no valor lido.

Uma desvantagem deste modelo é o fato dele ter sua vida curta quando exposto a um ambiente úmido por um longo período, para resolver este problema as hastes do sensor é revestido por um acabamento em outro. Outro detalhe que deve ser observado é a exposição do circuito eletrônico do sensor, o que pode comprometer as leituras.

Existem vários outros modelos no mercado que oferecem uma melhor resposta nas leituras de umidade além de também oferecer outros parâmetros como temperatura e acidez do solo que podem agregar ao projeto.

9 MATERIAIS E MÉTODOS

Com embasamento em todo o referencial teórico pesquisado para que este trabalho se concretizasse e se tornasse algo que respondesse aos propósitos, como uma solução para os problemas levantados e esperando um retorno satisfatório, alguns itens de *hardware* e de *software* foram definidos e apontados como o mínimo necessário para a criação de um protótipo de capacidade reduzida, que represente de forma similar a simulação de um projeto de grande escala.

9.1 Itens de *Hardware*

Os *hardwares* definidos para o desenvolvimento do protótipo do sistema de controle de irrigação, centralizou-se principalmente no Arduino. Devido ao número de portas de comunicação e os recursos de módulos e *Shields* que possibilita que o sistema funcione.

Foram utilizados dois modelos de Arduino o desenvolvimento deste trabalho. O Arduino principal que estará centralizado como um servidor para a aplicação é o modelo *MEGA* que contempla a quantidade de portas de entrada e saída de dados e o outro será um Arduino modelo *NANO* que tornará possível, através dos transceptores nRF24L01+ criar um enlace de comunicação sem fio entre Arduino *MEGA* e o Arduino *NANO*, eliminando os cabos necessários para ligar os sensores de umidade. Para possibilita-lo conectar a uma rede de comunicação Ethernet, utiliza-se o módulo ESP8266 ESP-07S.

Este trabalho consiste em duas áreas de atuação no sistema de irrigação, uma é a parte de sensoriamento e a outra são os atuadores. Para a medição da umidade do solo, os sensores disponíveis são vastos, o que diferencia um do outro é a composição dos materiais que eles são fabricados, como forma de proteção para um produto que ficará diretamente em contato com o solo e se degrada com o tempo. Outro fator é a precisão da leitura, o que impacta diretamente no custo deste sensor. Para este protótipo utiliza-se um sensor de baixo custo, o Higrômetro, que apresenta uma boa resposta para sistema de prototipagem. Já para a área de atuadores, como existe uma grande variedade de formas de irrigar o solo, este trabalho utiliza como principais elementos, o módulo de relés de até 8 canais que permite controlar cargas de 220 de VAC e corrente máxima de 10 ampères (A), podendo acionar contatores de grande capacidade ou pequenas motobombas. Em sistemas de irrigação por gotejamento, utiliza-se neste trabalho a válvula solenoide que controla a vazão da água em sistema de irrigação por gotejamento.

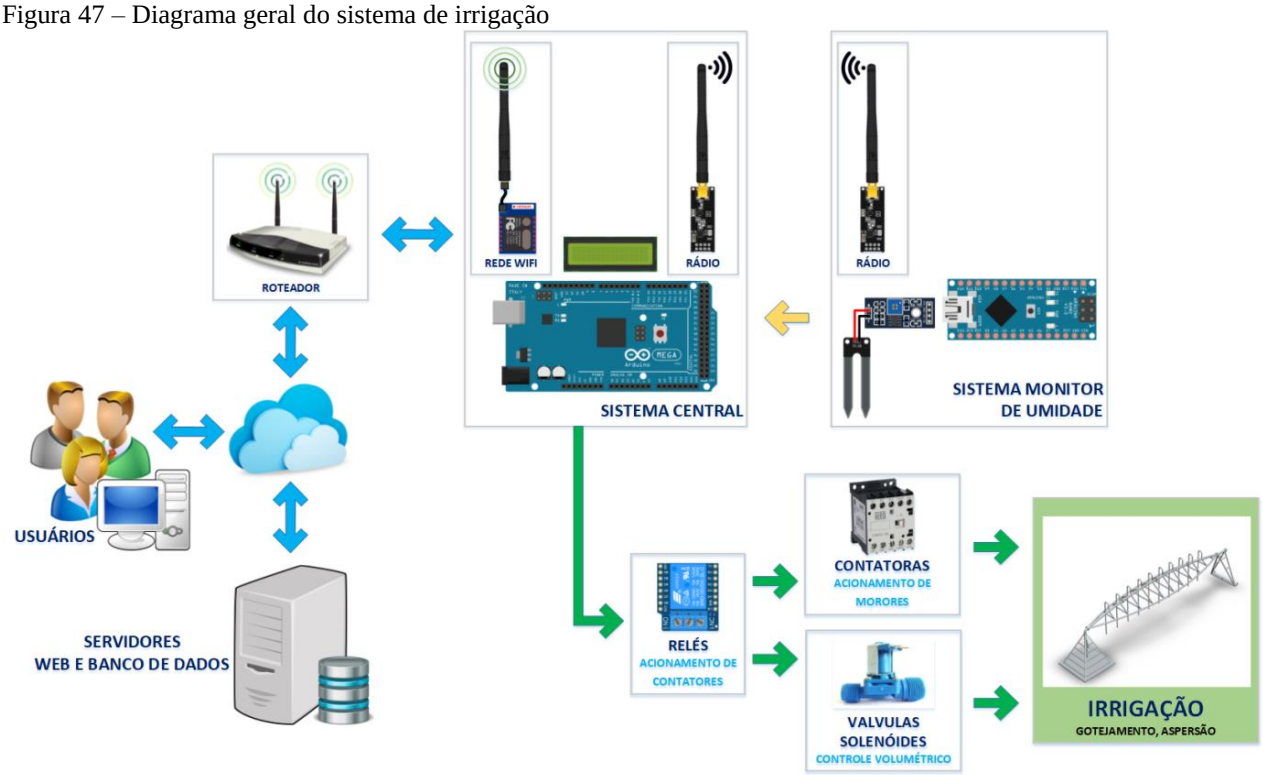
Para que todo sistema funcione, muita das vezes em ambientes onde a utilização de rede elétrica não esteja disponível, sugere-se a utilização de um sistema fotovoltaico isolados da rede ou autônomos, também conhecido como *OFF-Grid*, devidamente projetado para esta aplicação.

Outro componente importante para que este sistema de controle fique instalado em ambiente externo e protegido das ações do tempo, indica-se a utilização de caixas industriais de alta resistência e vedação, com grau de proteção IP66 ou IP67, conforme Resource Supply, LLC (2008), é o sistema de avaliação de proteção *Ingress Protect* (IP) que é um padrão definido pela norma internacional IEC 60529. O sistema de avaliação classifica o grau de proteção fornecido por um compartimento de equipamento elétrico contra objetos sólidos (como poeira) e líquidos (água, óleo, etc.).

9.1.1 Diagrama esquemático do projeto

A Figura 47, ilustra o diagrama geral do sistema de irrigação deste projeto, que compõe dos principais *hardwares* descritos acima e a forma como as informações são trocadas entre os componentes.

Figura 47 – Diagrama geral do sistema de irrigação

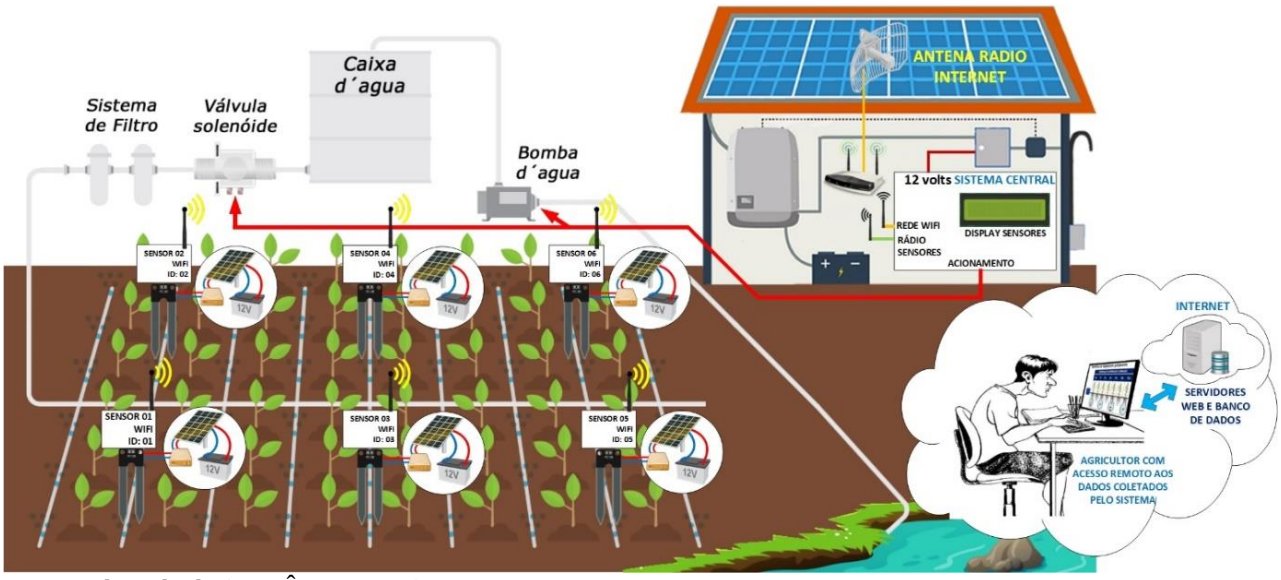


Fonte: O autor.

Com todo os sistema definido, passou-se ao detalhamento de cada parte do sistema para detalhar as particularidades importantes para o seu funcionamento correto. Dentre este detalhe destaca-se os níveis das tensões de alimentação dos módulos do Arduino e também da parte de comunicação entre estes componentes e o microcontrolador.

O mesmo sistema da Figura 47, inserido em uma imagem de situação em campo ficaria representado conforma a Figura 48, mostrando sua estrutura de trabalho.

Figura 48 – Diagrama geral do sistema de irrigação apresentado em campo.

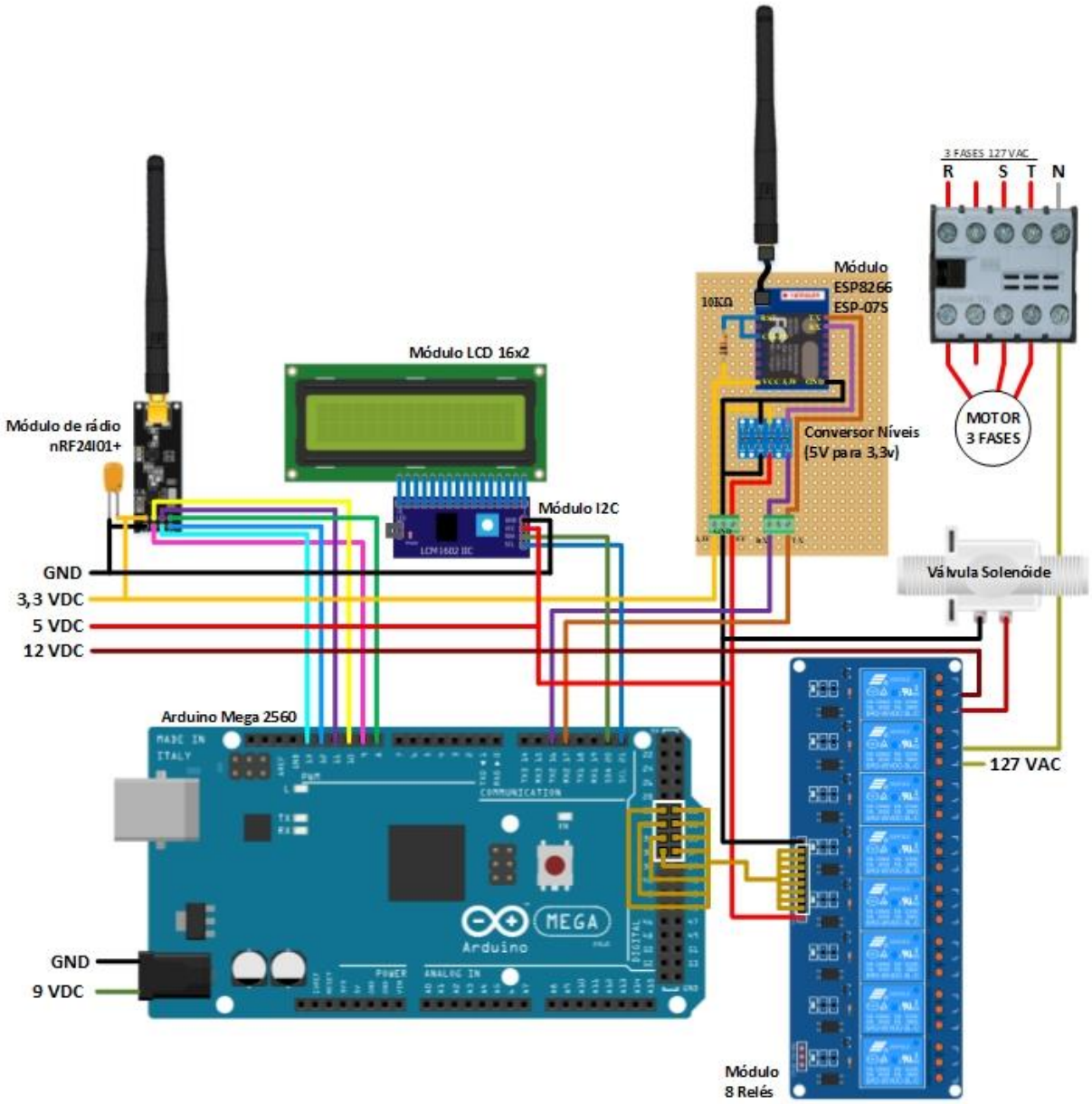


Fonte: Adaptado de (ASCÂNIO, 2010).

9.1.2 Sistema central do controle de irrigação

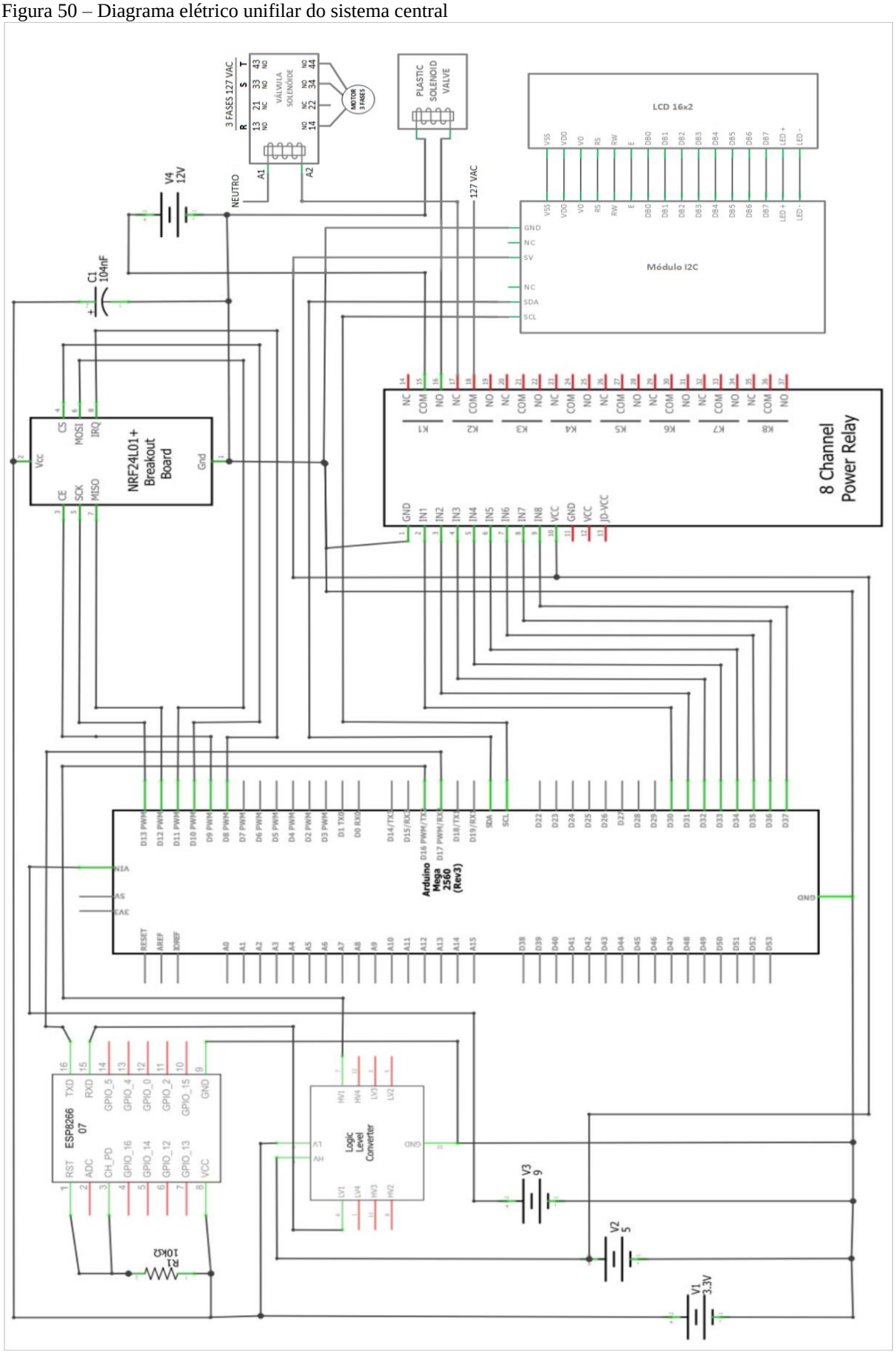
O sistema central é o que faz todo o controle e engloba a parte de recepção dos dados fornecidos pelo sistema monitor de umidade através da comunicação via rádio, ele também é responsável pelo acionamento dos dispositivos, através de relés e contatores, como motobombas, válvulas solenoides e outros dispositivos que venham a acionar automaticamente o sistema de irrigação, além de que o sistema central está conectado via WIFI à uma rede de Internet onde registra as leituras dos sensores de umidade em banco de dados para consultas de qualquer localidade através de uma página WEB. Este sistema central também possui um *display* de LCD de 16 colunas por 2 linhas que mostra em tempo real o valor que o sensor de umidade está medindo, seja de um *Pipe* apenas ou de um *Cluster* de seis *Pipes*. A Figura 49 mostra os componentes do sistema central e a Figura 50 o seu diagrama elétrico unifilar.

Figura 49 – Sistema Central de monitoração de umidade



Fonte: O autor.

Figura 50 – Diagrama elétrico unifilar do sistema central



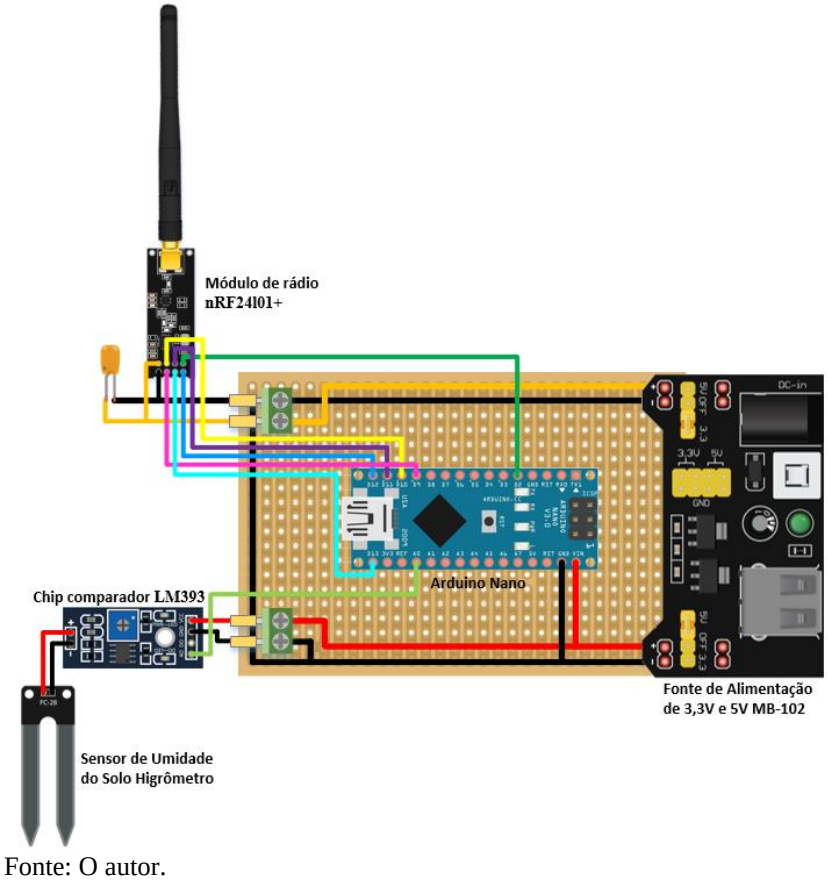
Fonte: O autor.

9.1.3 Sistema monitor de umidade

Esta parte do sistema de controle de umidade é de fundamental importância para o funcionamento do projeto. É através deste arranjo de *hardware* que a umidade do solo é monitorada, através do sensor higrômetro, entregue ao Arduino *NANO* que está em comunicação constante com o sistema central que irá atuar conforme as configurações estabelecidas para cada setor agrícola. A Figura 51, ilustra a composição do sistema monitor de umidade e a Figura 52 as ligações elétricas para o seu funcionamento.

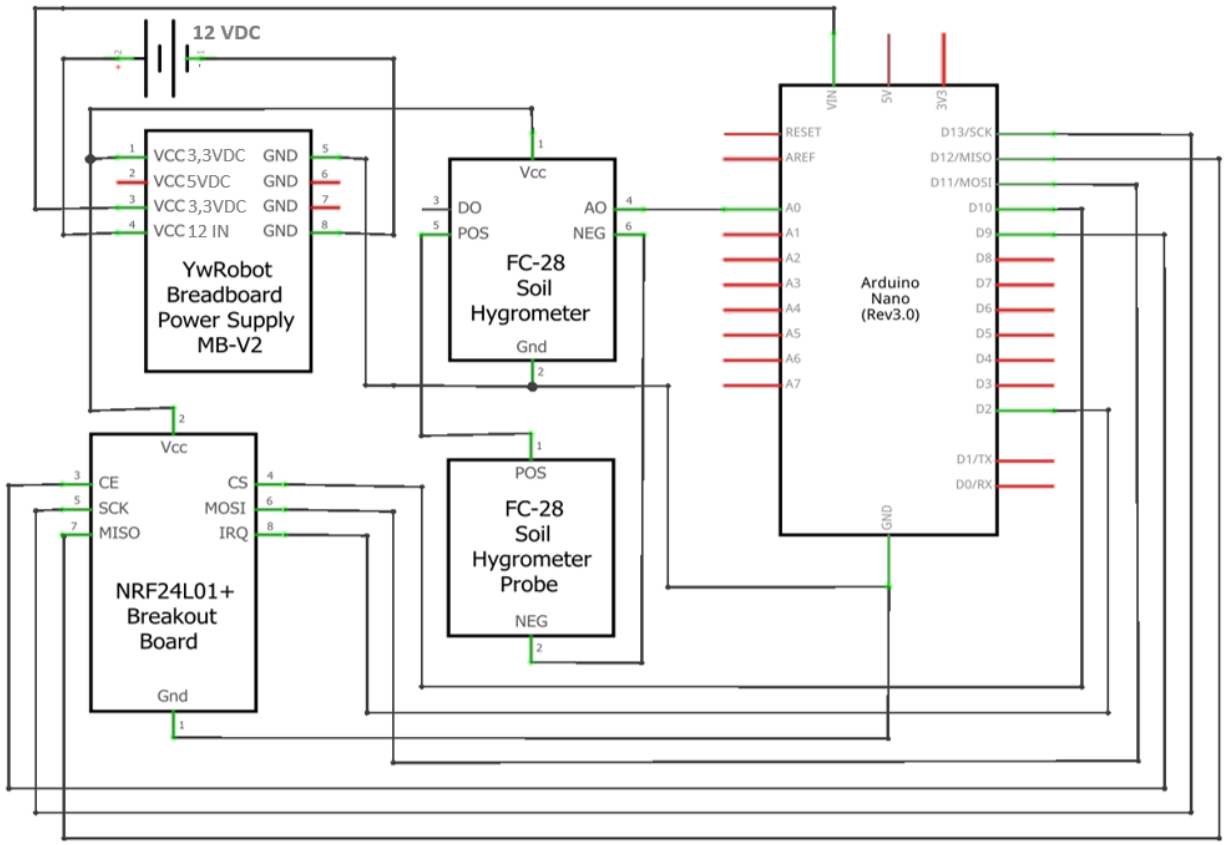
Com o uso do rádio nRF24l01+ é possível utilizar até seis conjuntos destes, respondendo ao sistema central, espalhados por uma área que pode ser pré-determinada pelo agricultor, dependendo da sua necessidade. Neste projeto por ser tratar de um protótipo, usou-se apenas um sistema monitor fazendo a leitura do solo e passando os dados obtidos numa frequência de leitura a cada 5 minutos. O sensor higrômetro e o Arduino *NANO* são alimentados com 5 *Volts*, já o módulo de rádio nRF24l01+ trabalha com 3,3 *Volts*. Ambas as tensões são provenientes do módulo MB-102 que oferece na sua estrutura estes dois níveis o que facilitou a montagem.

Figura 51 – Diagrama do sistema monitor de umidade



Fonte: O autor.

Figura 52 - Diagrama elétrico unifilar do monitor de umidade

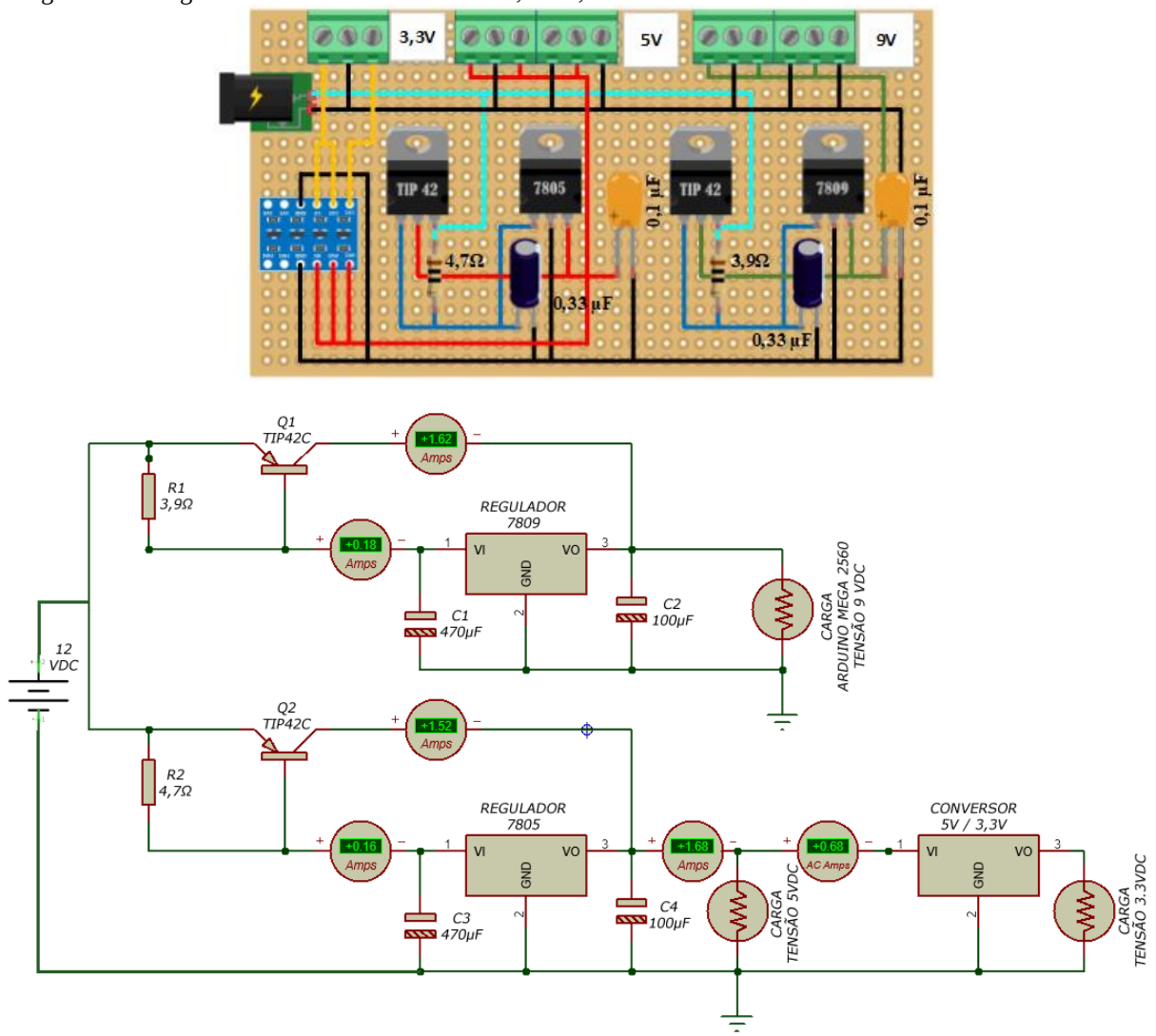


Fonte: O autor.

9.1.4 Alimentação do sistema

O sistema central é composto por vários módulos e alguns deles, como o rádio nRF24l01+ e o módulo de WIFI ESP8266 ESP07 trabalha com o nível de tensão na recepção de dados de 3,3 *Volts* tensão contínua e a alimentação do Arduino, conforme Souza (2014), pode trabalhar com valores de tensão da fonte externa entre os limites de 6 *Volts* a 20 *Volts*, porém se alimentar com uma tensão abaixo de 7 *Volts*, a tensão de funcionamento do Arduino de 5 *Volts* pode ficar instável e quando alimentada com tensão acima de 12 *Volts* o regulador de tensão da placa pode sobreaquecer e danificar o microcontrolador. Desta forma o recomendado é que a tensão de alimentação deve estar compreendida de 7 *Volts* à 12 *Volts*. Partindo desta informação necessitou-se desenvolver um circuito eletrônico com reguladores de tensão que atendesse o funcionamento do projeto nas tensões de 3,3 VDC, 5 VDC e definida a tensão de 9 VDC para o Arduino *MEGA* 2560, conforme a Figura 53. Como a corrente nominal dos reguladores é no máximo de 1 ampère, fez-se o uso de transistores de potência, também conhecidos como TIP, modelo 42C, que apresenta um ganho na potência fornecida pelo circuito.

Figura 53 – Diagrama do circuito eletrônico de 9, 5 e 3,3 *Volts*



Fonte: O autor.

Os valores dos capacitores C1, C2, C3 e C4 dos reguladores de tensão de 9 *Volts* e de 5 *Volts* respectivamente foram especificados pelo fabricante em seu *Datasheet*, onde C1 e C3 se faz necessário caso esteja distante do filtro e C2 e C4 eliminam ruídos em alta frequência e melhora a estabilidade da saída, conforme Fairchild Semicondutores (2015).

Para o cálculo da corrente nestes reguladores que não possa atingir o seu valor nominal máximo de trabalho, calculou-se o valor de R1 e R2.

$$I = \frac{VBE}{R} = \frac{0,7}{3,9} = 179\ mA \qquad (02)$$

Segundo Silva (2015), Como o *Base-Emissor Voltage* (VBE) do transistor é tipicamente 0,7 *Volts* o transistor será acionado se a corrente de carga for igual ou superior a 179 mA.

Para o regulador de 5 *Volts* o cálculo ficou da seguinte forma:

$$I = \frac{VBE}{R} = \frac{0,7}{4,7} = 149\ mA \quad (03)$$

Com este resultado, a saturação do transistor se dá com a corrente superior a 149 mA.

9.2 Itens de *Software*

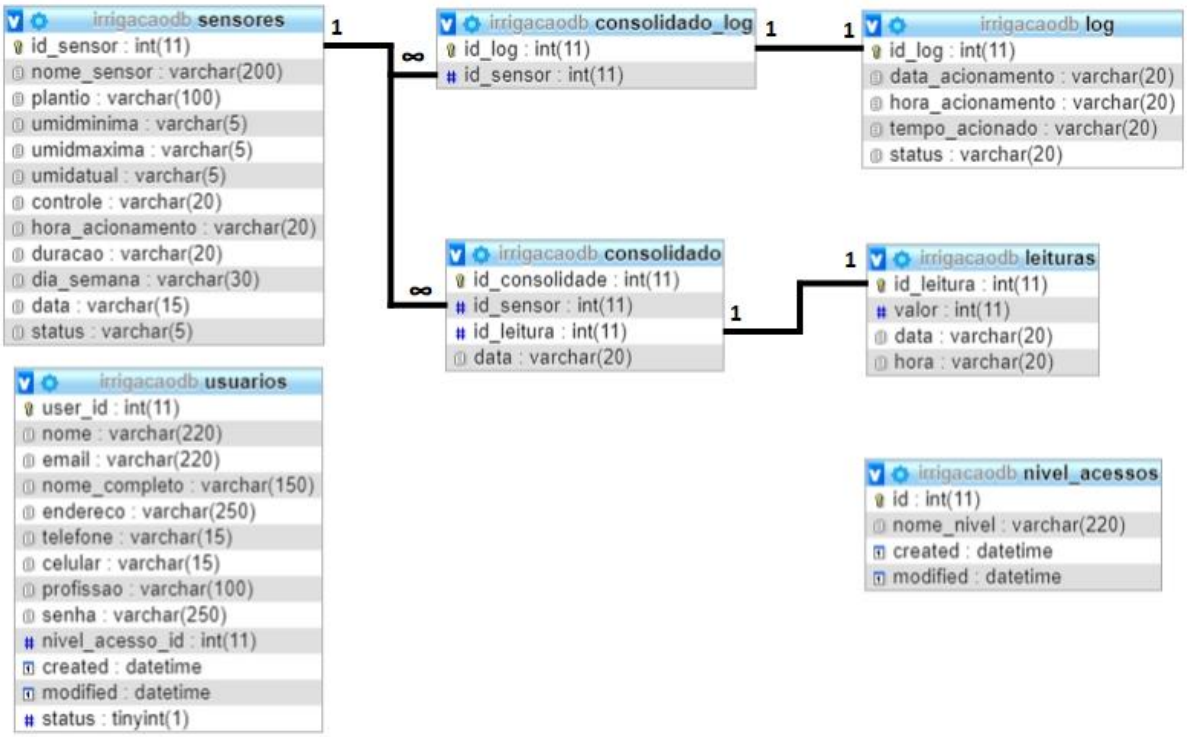
Vários itens de *software* são necessários para se criar um sistema de monitoração de umidade e disponibilizar seus dados para acompanhamento. O primeiro a ser definido é o sistema por onde o operador terá acesso as configurações de funcionamento e leitura dos dados armazenado acerca de um ponto de atuação no campo, definiu-se a linguagem de programação PHP e banco de dados o MySQL, por onde o operador irá fazer o acesso aos registros dos valores lidos pelos sensores em campo e disponibilizados em uma página *WEB* juntamente com os interpretadores de *Scripts* como o *JavaScript*, *Bootstrap*, etc.

Outro *software* importante é o IDE do Arduino. *Software* fornecido para uso livre e essencial para a criação dos *Sketchs*, códigos de programação que são carregados para a memória do *hardware* Arduino para que ele se comporte e trabalhe conforme a necessidade do programador. Através dele é possível carregar todas as bibliotecas para o funcionamento dos módulos e *Shields* do Arduino.

9.2.1 Sistema de monitoração *WEB*

Para o desenvolvimento do sistema por onde o usuário possa ter uma monitoração do sistema de irrigação remotamente e saber com qual frequência o sistema está sendo acionado, definiu-se a base de dados (BD) por onde todas as informações do sistema ficam armazenadas, nela tem-se o cadastro dos usuários que são habilitados a acessar à página *WEB*, o cadastro dos sistemas de sensores juntamente com as principais características de acionamento, monitoração diária das leituras que o sistema em campo realiza além do *log* das operações. A Figura 54, mostra a estrutura do BD e também o relacionamento entre as tabelas, que tornam o sistema mais dinâmico e organizado.

Figura 54 – Estrutura do banco de dados e seus relacionamentos



Fonte: O autor.

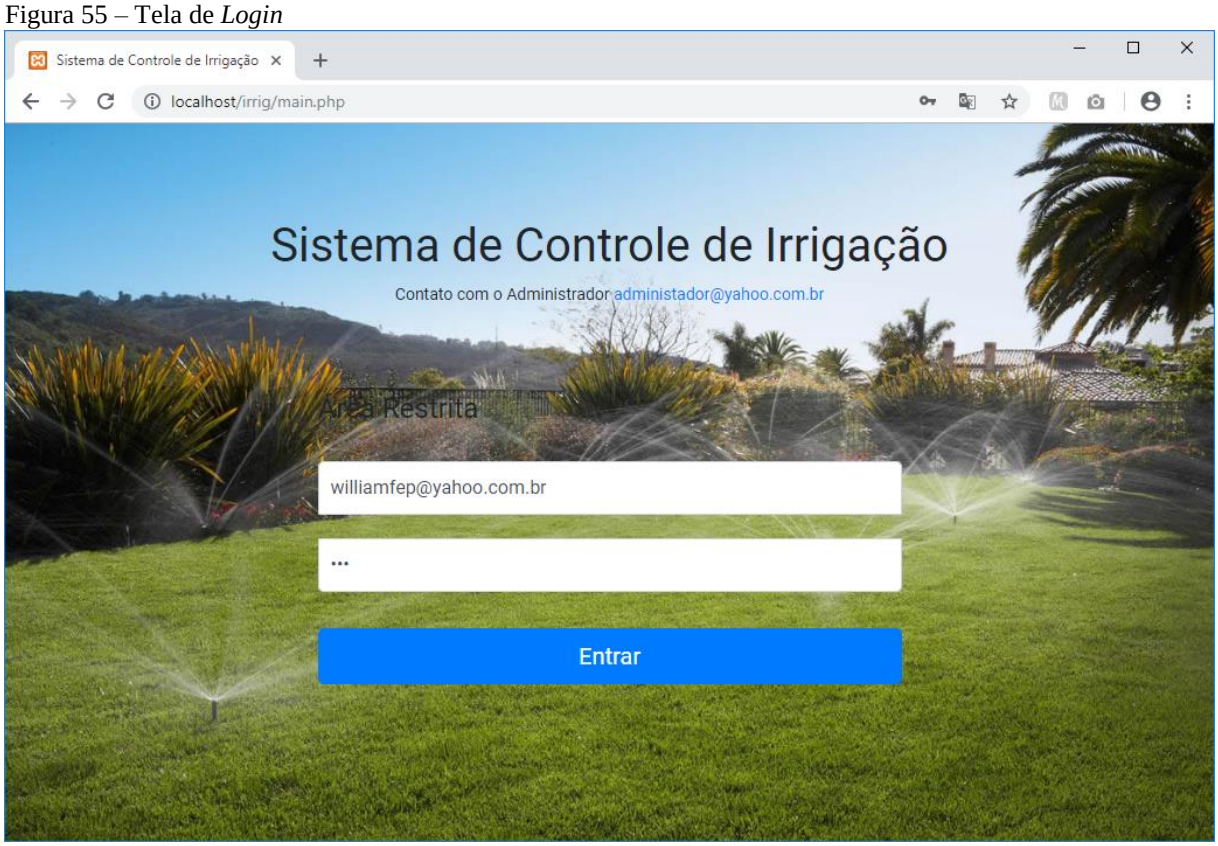
Com o banco de dados definido, desenvolveu-se a página *WEB* que é a ferramenta fundamental para o funcionamento do sistema, onde os usuários cadastrados podem acessar de qualquer dispositivo, seja um computador, *Tablet* ou *Celular* e cadastrar e monitorar os pontos de irrigação.

A página *WEB* para que possa ser acessada de qualquer local que ofereça uma conexão de internet, primeiramente ela precisa estar armazenada em um provedor *WEB*, onde segundo Toor Tecnologia (2017), um o provedor de internet é um serviço online, administrado por uma empresa específica, que oferece algumas ferramentas básicas como hospedagem de *sites*, servidor de *e-mails*, mecanismos de buscas, armazenamento em nuvem etc. Além desta hospedagem, todo site necessita de um endereço para que seja acessado, este endereço é conhecido como *Internet Protocol* (IP) e através deste IP é que são criados os domínios, que segundo *Verisign* (2018), um domínio é um nome dado para um IP para que uma página seja identificada na Internet. Este domínio pode ser registrado em vários órgãos responsáveis, mas o mais conhecido é o Registro.br acessado pelo site www.registro.br.

Como este trabalho oferece apenas um protótipo do sistema de irrigação, a página *WEB* criada é acessada através de um servidor local, dispensando a necessidade do uso de domínios, sendo acessada diretamente pelo IP do computador ou pelo nome *localhost* que representa o acesso local.

A Figura 55 apresenta a tela de *Login* do sistema de controle e monitoração.

Figura 55 – Tela de *Login*



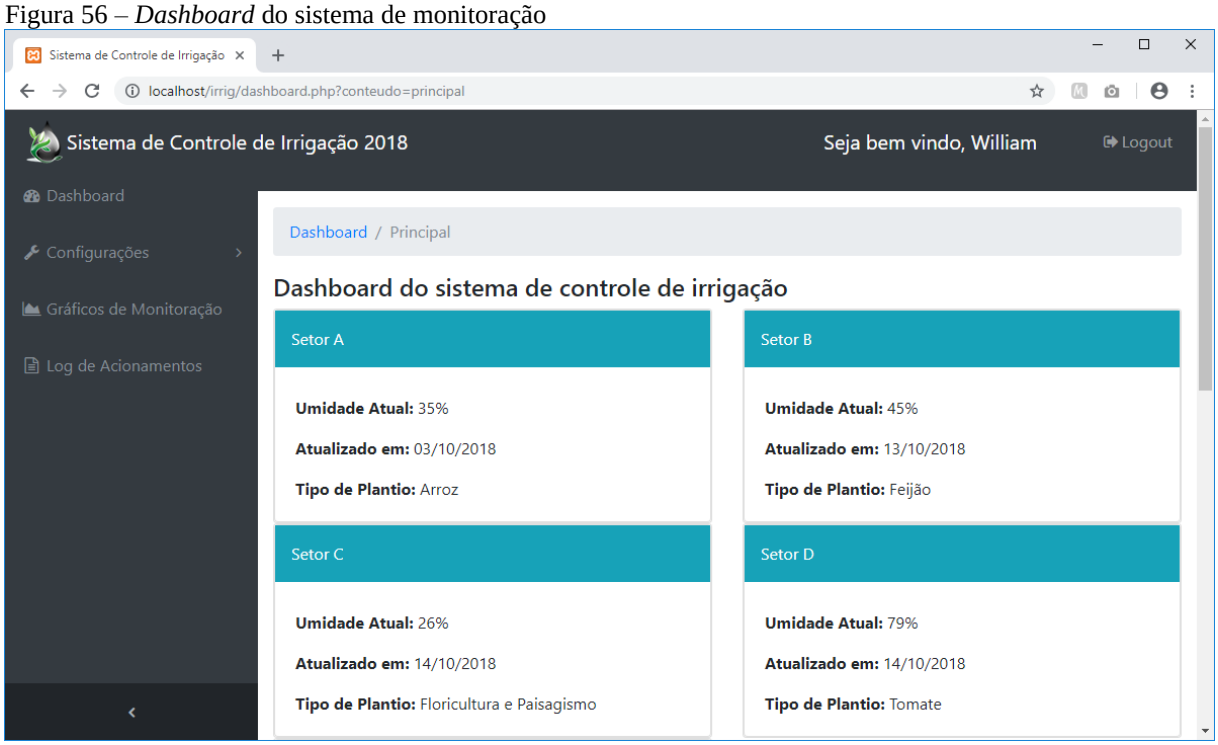
Fonte: O autor.

Com os usuários previamente criados pelo administrador do sistema, é possível através do *e-mail* e de uma senha acessar o sistema. Sem estas informações o usuário não terá acesso.

Após fazer o *Login* no sistema, o usuário é direcionado para a página principal denominada *Dashboard*, que segundo Nascimento (2017), um *Dashboard* é um painel que mostra as principais informações do sistema, onde o usuário já visualiza a métrica e indicadores importantes de forma visual, facilitando a compreensão das informações de um sistema.

A Figura 56 apresenta a *Dashboard* do sistema de monitoração

Figura 56 – *Dashboard* do sistema de monitoração

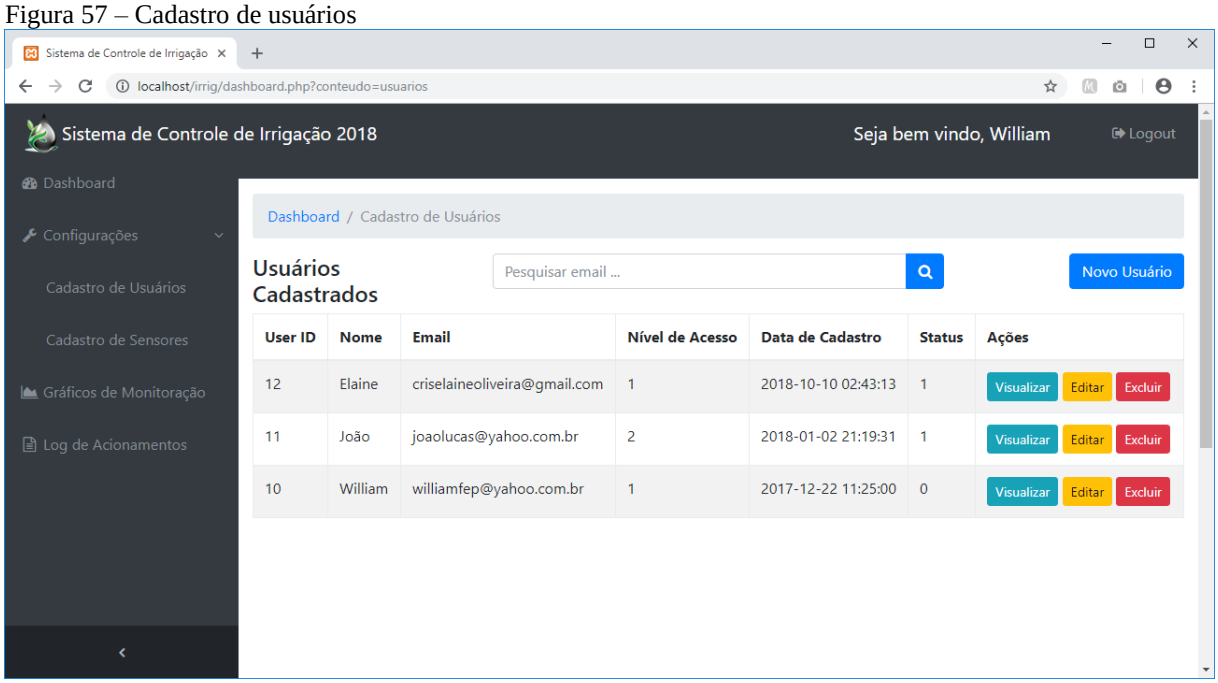


Fonte: O autor.

Através desta tela já se pode visualizar os sensores cadastrados e obter algumas informações, como a Umidade Atual, a Data da última atualização e o tipo de Plantio. Além destas informações a tela apresenta um *menu* de acesso à outras funcionalidades como o Cadastrado de Usuários, Cadastro de Sensores, o Gráfico de Monitoração e *Log* de Acionamentos.

Como primeira medida, o administrador cadastra os usuários através do *menu* Configurações / Cadastro de usuários, onde a lista de usuários cadastrados é exibida, conforme a Figura 57, tendo este a opção de cadastrar um novo usuário, visualizar outros detalhes deste usuário, editar ou excluir as informações do usuário.

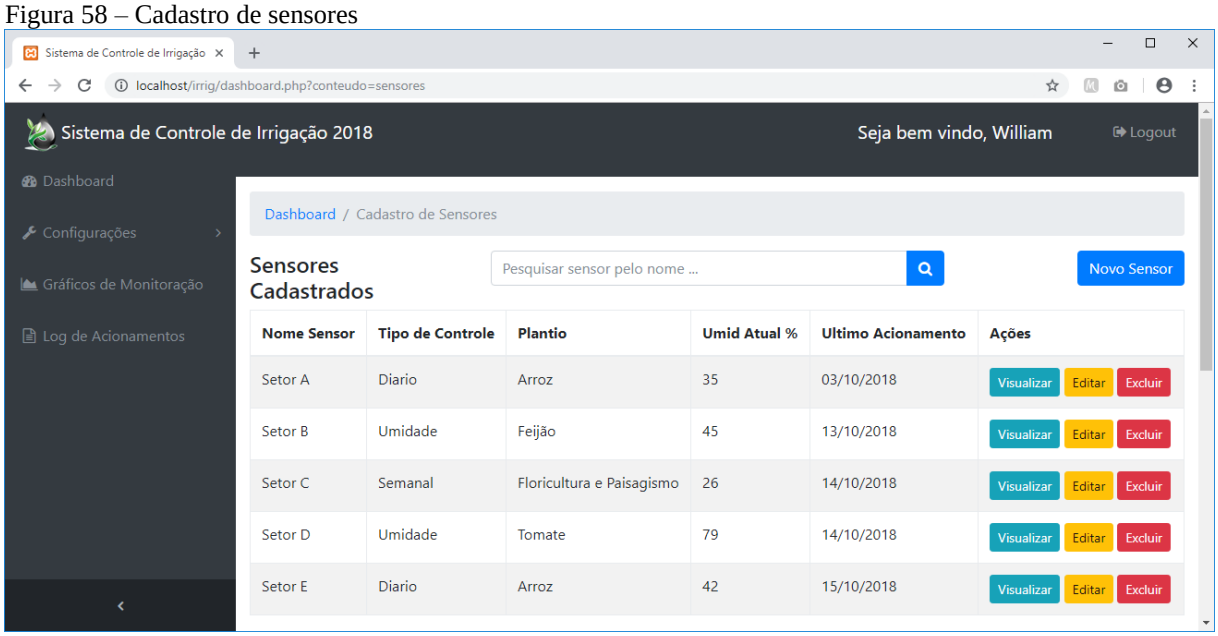
Figura 57 – Cadastro de usuários



Fonte: O autor.

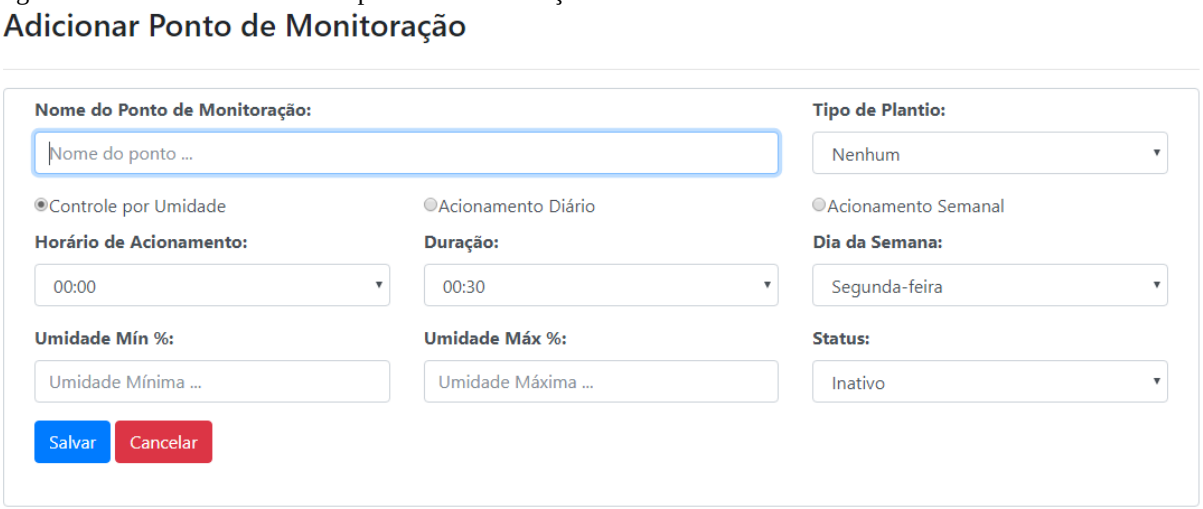
O *menu* para o cadastro de sensores, pode ser visto na Figura 58. Neste o usuário pode visualizar a lista de sensores cadastrados, fazer uma busca por sensores, fazer novos cadastros e também visualizar os outros detalhes além de poder editar ou remover este sensor. As principais informações que devem ser digitadas para o cadastro do ponto de monitoração são mostrados na Figura 59.

Figura 58 – Cadastro de sensores



Fonte: O autor.

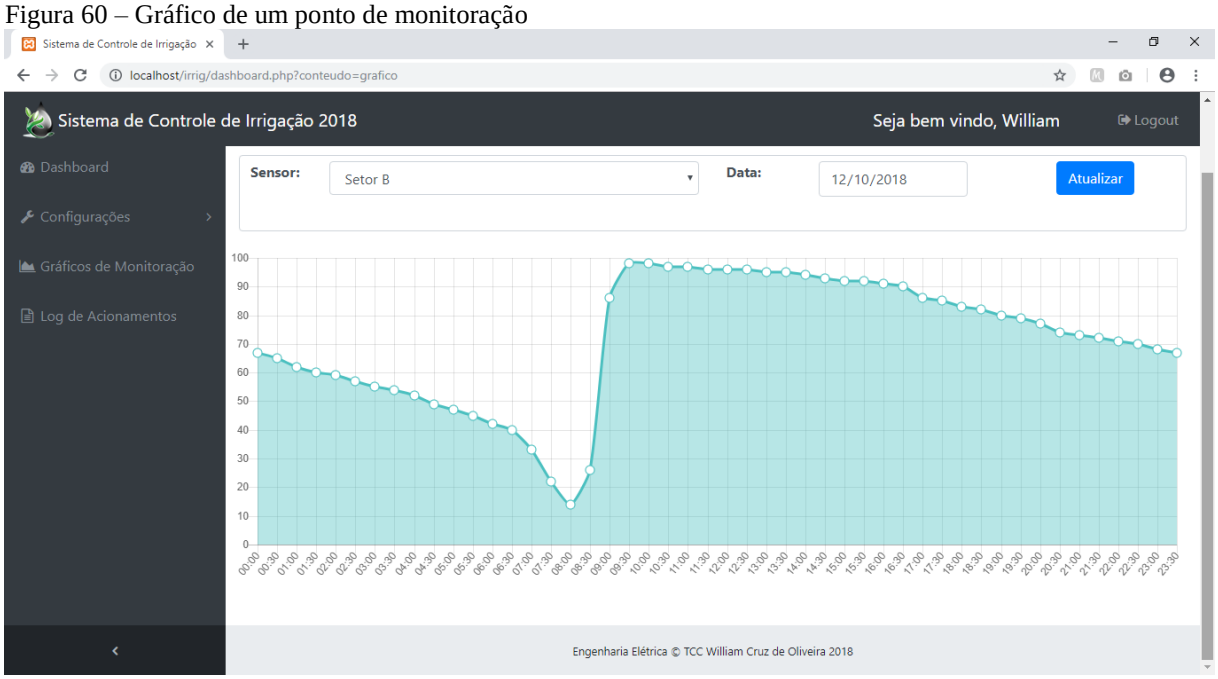
Figura 59 – Cadastro de um novo ponto de monitoração



Fonte: O autor.

Para o gráfico de monitoração, desenvolveu-se uma página onde é possível o usuário monitorar os valores de umidade de um determinado ponto de monitoração durante um dia e saber dos acionamentos do sistema de irrigação, conforme a Figura 60.

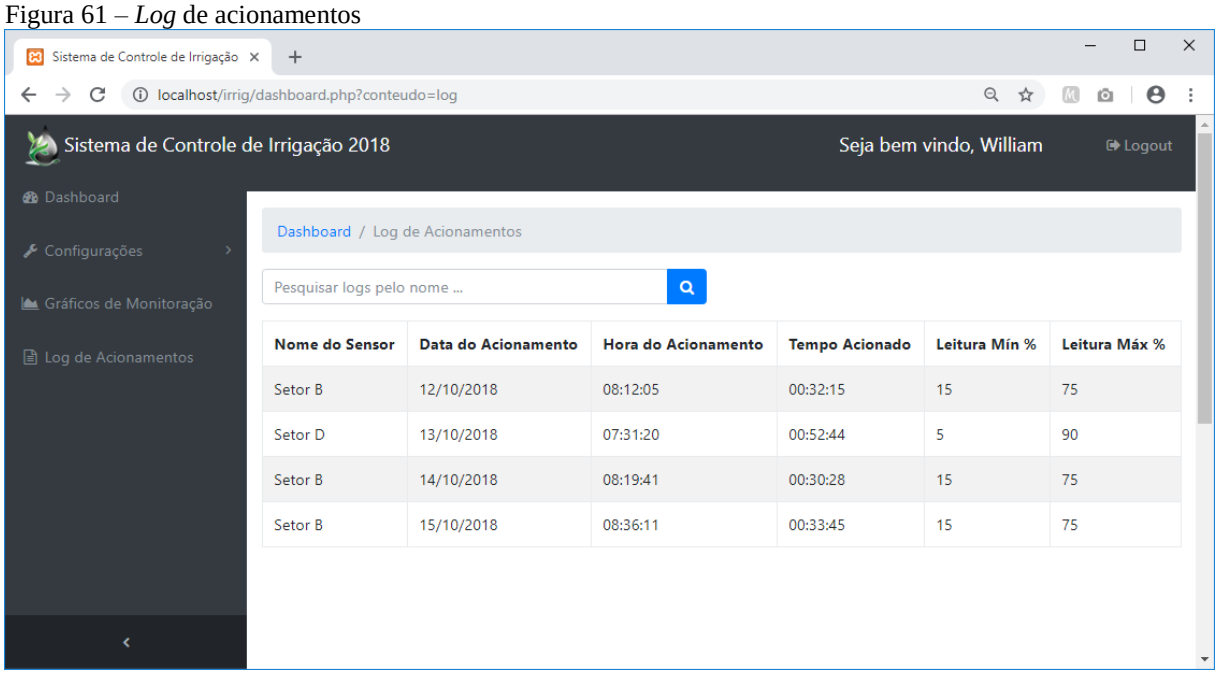
Figura 60 – Gráfico de um ponto de monitoração



Fonte: O autor.

Para concluir o sistema de monitoração, foi disponibilizada uma página onde o usuário pode acompanhar a operação de cada sensor, saber a data do último acionamento, a hora do acionamento, o tempo que o sistema ficou acionado, além da leitura mínima e máxima que foi definido para acionamento, conforme a Figura 61.

Figura 61 – *Log* de acionamentos



Fonte: O autor.

9.2.2 IDE Arduino

Os microcontroladores utilizados neste trabalho, o Arduino *MEGA* para o Sistema Central do controle de Irrigação e o Arduino *NANO* para o sistema monitor de umidade, por si só não são capazes de executar nenhuma função, suas portas de *Input* e *Output* ficam inativas e sua capacidade de processamento inutilizada. Para isso é necessário a criação de algoritmos que são elaborados no IDE do Arduino, compilados e carregados para a memória destes microcontroladores para que estes dispositivos trabalhem conforme a nossa vontade. Este algoritmo para Arduino denominado *Sketches* utilizam a criatividade dos desenvolvedores, mas possuem bibliotecas padrões para cada módulo e *Shield* utilizado no Arduino. No ANEXO A apresenta-se o algoritmo do sistema central de controle de irrigação e no ANEXO B apresenta-se o algoritmo do sistema de monitoração, incluindo as bibliotecas de cada fabricante e dispostos a trabalhar para tornar meros módulos em dispositivos capazes de fazerem leituras de sensores, comunicarem através dos rádios, atuar no relé e disponibilizar todas as informações relevantes, à um servidor, por onde agricultores e administradores rurais possam monitorar o sistema que se torna automatizado.

Para o correto funcionamento do sistema de monitoração de umidade, dois algoritmos, *Sketches* foram criados. Um deles é para o sistema principal e o outro para o sistema de monitoração da umidade.

A estrutura principal como o próprio nome o caracteriza, desenvolveu-se para que todo o sistema funcione, cumprindo o propósito deste trabalho. É através deste algoritmo que o Arduino comunicará com os seus *Shields* e cada um irá executar a sua função. O código fonte criado engloba as bibliotecas do módulo de LCD, do módulo de comunicação WIFI, do módulo de rádio e do controle do módulo de relé que irá acionar o (s) contator (es) e válvula (s) solenoide (s).

Já o *Sketch* do sistema de monitoração da umidade que fica ligado diretamente ao sensor, possui um algoritmo desenvolvido especificamente para acompanhar as leituras do sensor higrômetro e transmiti-las para o sistema principal que tomará a decisão conforme a necessidade criada para cada caso.

10 MONTAGEM DO PROTÓTIPO

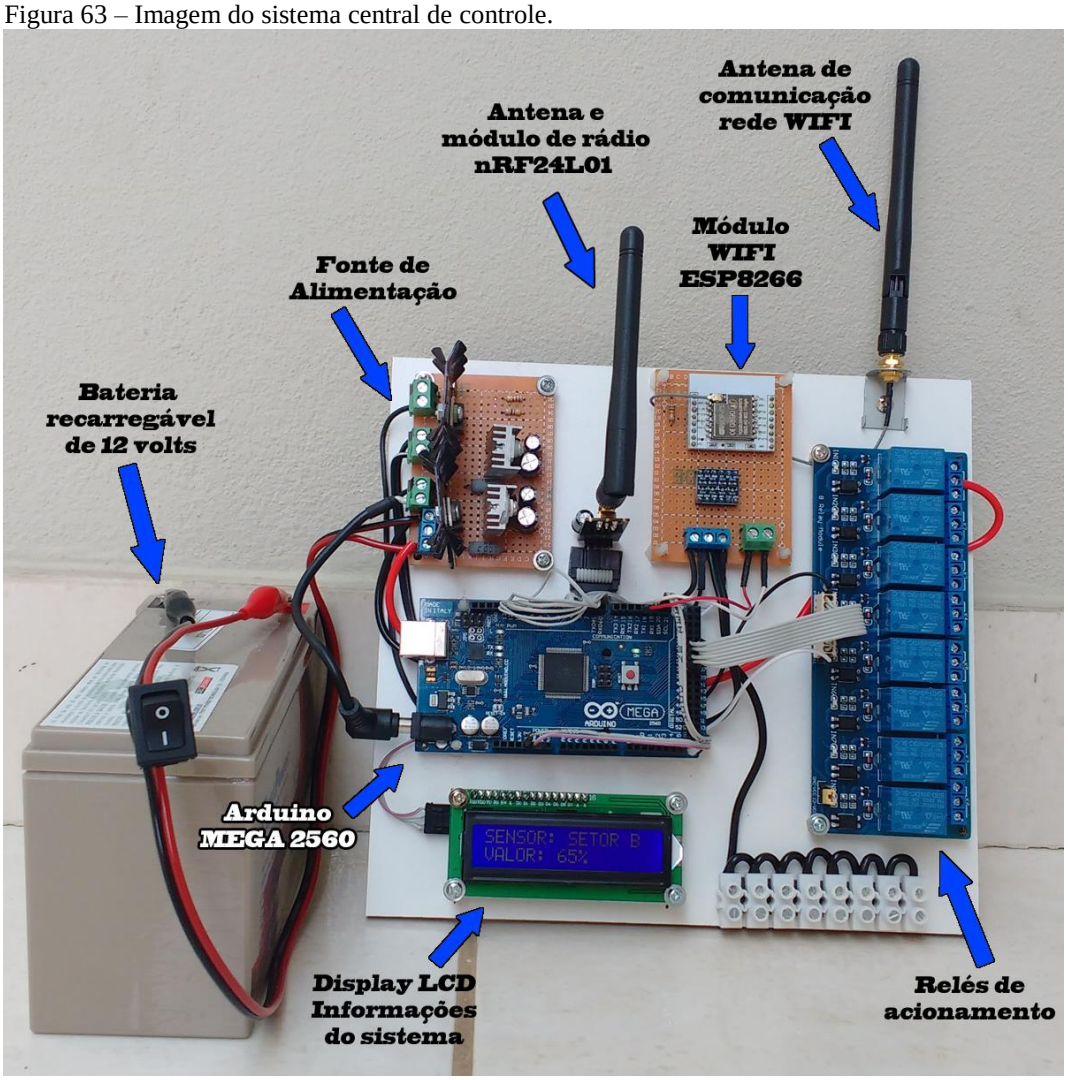
Com os módulos e *Shields* definidos para a montagem do protótipo, algumas características peculiares entre estes componentes eletrônico fez-se necessário o desenvolvimento de uma placa eletrônica que tivesse 12 VDC de tensão de entrada proveniente de um sistema fotovoltaico sugerido, porém três tensões específicas eram necessárias na saída para alimentação do sistema, 9 *Volts*, 5 *Volts* e 3,3 *Volts*, conforme a Figura 62.

Figura 62 – Fonte de tensões de 9, 5 e 3,3 *Volts*.

Fonte: O autor.

O sistema central de controle foi montado sob uma plataforma de madeira para melhor organizar os módulos, *Shields* e o Arduino. Desta forma, ele pode ser montado em caixas industriais de alta resistência e vedação conforme já mencionado. A Figura 63 ilustra o sistema principal.

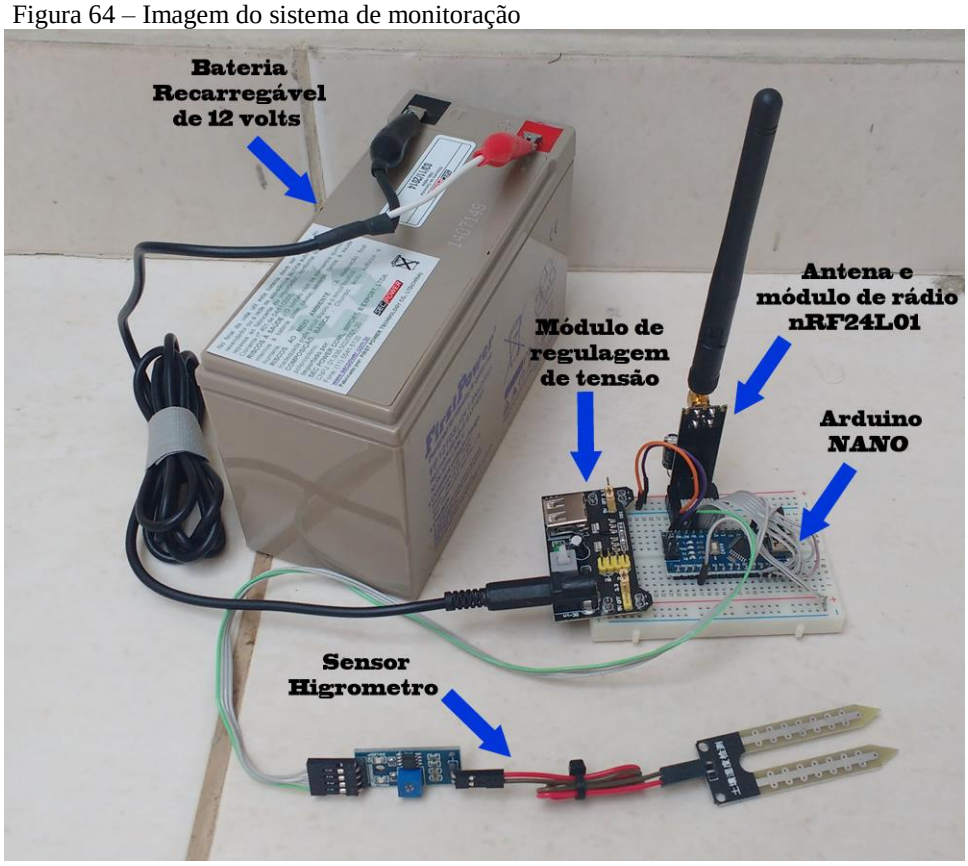
Figura 63 – Imagem do sistema central de controle.



Fonte: O autor.

Já o sistema de monitor de umidade, por se tratar de um protótipo, foi montado em um *pront-o-board*, conforme a Figura 64, que oferece a opção de realizar a montagem e testar para posteriormente confeccionar uma placa de circuito impresso e disponibilizar uma placa definitiva.

Figura 64 – Imagem do sistema de monitoração



Fonte: O autor.

10.1 Protótipo em operação

Este trabalho foi elaborado para que houvesse uma interação entre o agricultor e o sistema de forma indireta e com tratativas precisas que aperfeiçoasse o trabalho de irrigação em qualquer tipo de plantação. Para isso o sistema foi dividido com ferramentas que estivessem na mão deste trabalhador e também atuando no campo.

Com a elaboração do acesso via internet, qualquer dispositivo conectado à rede pode oferecer o acesso ao sistema e é de fácil manuseio. Para o ajuste do sistema em campo, alguns parâmetros e testes foram realizados para homologar um sistema, que está em um formato simplificado que neste trabalho tratamos por protótipo.

10.1.1 Ajuste do sensor higrômetro

Por ser tratar de um sensor de baixo custo, desenvolvido para pequenas aplicações e trabalhos acadêmicos, este sensor de umidade higrômetro apresenta uma leitura, que segundo

Fritzen (2016), varia de 0 a 1023 mas apresenta variações que necessitam ser analisadas e obter o valor real em operação.

O ajuste é bem simples, são realizadas duas leituras do sensor, uma em estado totalmente seco, ou seja, fora do ambiente de monitoração onde pega-se o valor máximo, representando o solo em estado seco e posteriormente insere-se o sensor em um recipiente com água onde ira-se obter o valor mínimo que representa a umidade máxima que o sensor é capaz de identificar. Este processo ocorre com a variação de resistência no sensor, quanto menor a resistência, maior a umidade e quanto maior a resistência representa o solo seco.

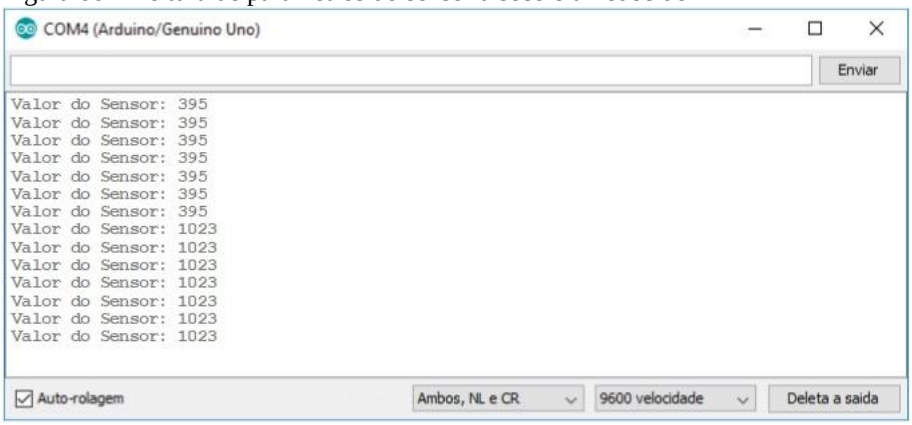
A Figura 65 ilustra os procedimentos realizados para a obtenção dos valores mínimo e máximo para serem definidos em algoritmos.

Figura 65 – Obtenção das leituras mínima e máximo do sensor higrômetro

Fonte: O autor.

Com este procedimento obteve-se o valor a seco de 1023 e com o sensor umedecido de 395, conforme a Figura 66. Estes valores foram passados para controle do algoritmo para um registro correto dos valores obtidos em campo.

Figura 66 – Leitura de parâmetros do sensor à seco e umedecido



Fonte: O autor.

A programação do Arduino oferece um comando interno ao IDE que baseado em dois valores ele entrega o resultado final em porcentagem, conforme a imagem 67.

Figura 67 – Comando do Arduino para conversão de grandezas

```
valor = map(valor_analogico, vminimo , vmaximo, 0, 100);
```

Fonte: (ARDUINO.CC, 2017)

Sintaxe do comando *map*:

a) *valor* = Variável que recebe o valor convertido em porcentagem.
b) *valor_analogico* – Valor lido pelo sensor, que varia de 395 à 1023.
c) *vminimo* – Valor definido de 395, conforme análise.
d) *vmaximo* – Valor definido de 1023, conforme análise.
e) *Valor 0* – Mínimo em porcentagem.
f) *Valor 100* – Máximo em porcentagem.

Com estes parâmetros definidos, a próxima etapa foi aplicar em campo para o desempenho do sistema e a aquisição de valores reais.

10.1.2 Experimento em campo

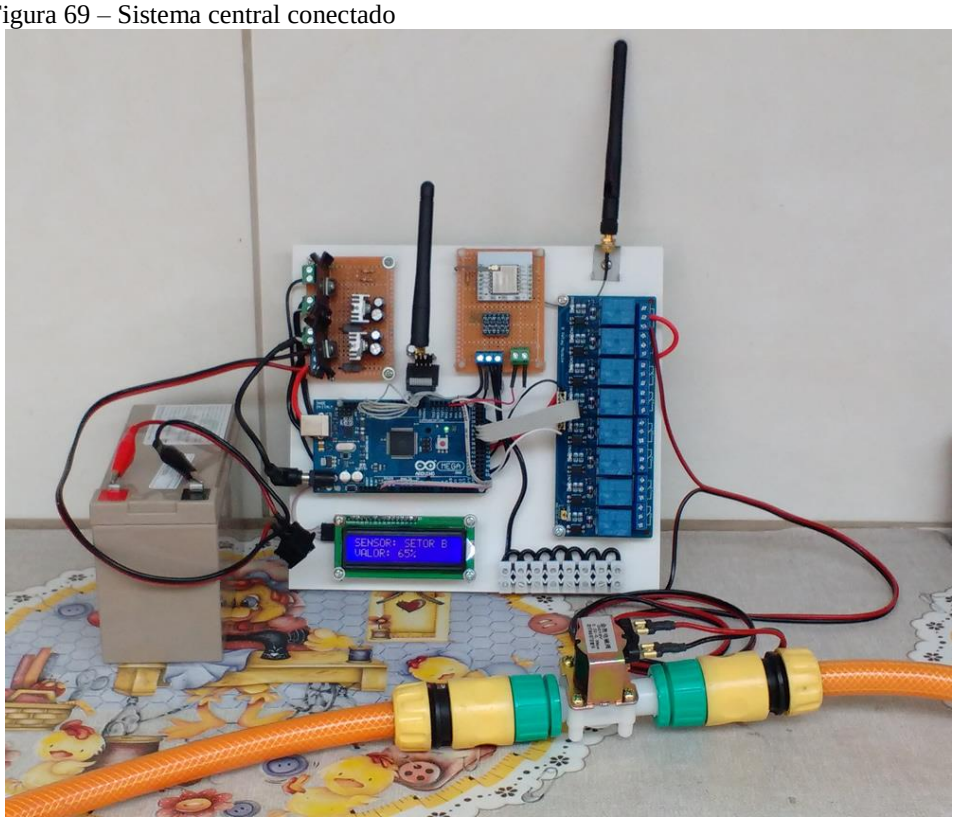
Depois de todo o sistema montado, foi preciso fazer o teste em campo para a homologação deste trabalho. Escolheu-se algumas flores dispostas em vasos e definiu-se o valor mínimo e máximo monitorando a umidade da terra nestes vasos durante um dia, onde a temperatura ambiente variou de 18º C e a máxima chegou a 27º C. O tempo estava limpo, sem chuva. A medição mínima de umidade da terra chegou a 31% (valor 552 no sensor) e foi definido 65% como o valor máximo de umidade nestas plantas. O sensor foi disposto em um dos vasos conforme a Figura 68 e posteriormente a parte eletrônica protegida para não se molhar com o acionamento do micro aspersor.

Figura 68 – Irrigação controlada em teste com flores em vasos

Fonte: O autor.

O sistema central ficou instalado em uma outra área, conforme a Figura 69, onde estava pronto para receber as leituras do sistema monitor via rádio e acionar a válvula solenoide quando atingisse o valor mínimo de umidade, iniciando a irrigação.

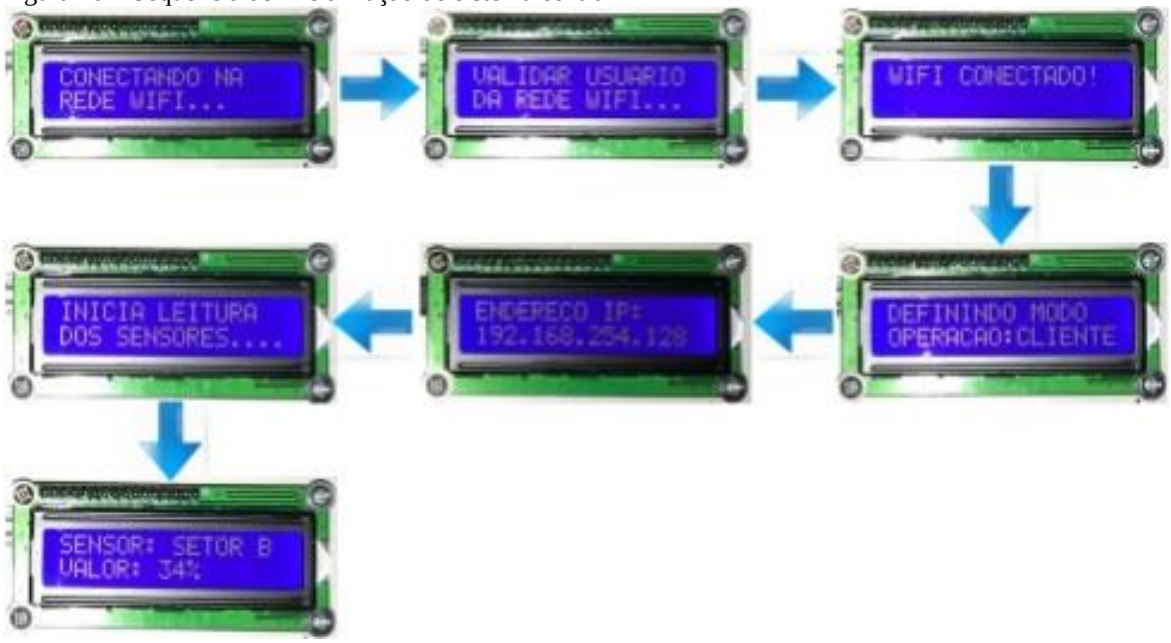
Figura 69 – Sistema central conectado



Fonte: O autor.

Toda a parte de inicialização do sistema central é acompanhado pelo *display* LCD e mostra as mensagens conforme inicializa e prepara o seu funcionamento do sistema. A Figura 70 mostra todos os passos que se inicia na tentativa de conexão com a rede WIFI, faz-se a validação, confirma a conexão, defina-se o modo de operação, obtém-se o endereço IP para conexão na Internet, informa o início das leituras pré-definidas em algoritmo, buscando no banco de dados as características do setor onde será a monitoração e inicia-se as leituras.

Figura 70 – Sequência de inicialização do sistema central



Fonte: O autor.

Após a inicialização o sistema passa a monitorar a umidade do solo conforme as configurações registradas no sistema *WEB*, registrando as leituras em intervalos de 30 minutos. A cada acionamento um *log* é registrado informando a data e a hora do acionamento e o tempo da irrigação. Desta forma o agricultor pode acompanhar como está o andamento do seu plantio.

11 CONCLUSÃO

O desenvolvimento deste trabalho possibilitou através dos estudos e da elaboração de um protótipo, comprovar que é possível através de recursos tecnológicos de baixo custo atender a demanda da maioria dos agricultores que hoje em dia utilizam a irrigação de maneira manual e descontrolada, o que traz inúmeros resultados ruins, seja na qualidade do produto ou no faturamento, mas o foco principal e o mais importante é com relação a responsabilidade com o meio ambiente.

Várias linhas de pesquisas foram criadas e muitas delas se divergiam em assuntos totalmente diferentes o que tornou este trabalho desafiador, onde a oportunidade de se fazer algo que realmente pudesse contribuir, mesmo de forma discreta, para mundo melhor sem desperdício de um recurso tão valioso que é a água.

Assim, buscou-se as tecnologias que estão em foco no mercado, sem fazer uso de componentes que onerassem o desenvolvimento do trabalho, mas que o tornasse útil, prático e eficiente.

A Iot, amplamente discutida entre os amantes de tecnologia, foi o alicerce para que este trabalho se concretizasse. Todos os componentes foram pensados e estruturados de forma a se comunicarem e agirem de forma inteligente, do ponto de vista computacional, para superar uma deficiência que é a irrigação feita de forma desordenada, trazendo vários prejuízos, tanto para o agricultor mas principalmente para o meio ambiente.

O microcontrolador Arduino possibilitou que toda esta linha de desenvolvimento se tornasse realidade, pela sua eficiência e por existir um infinito mercado de módulos e placas que o completam e nos possibilitam usar a imaginação, baseado nas necessidades. Assim usou-se sensores de baixo custo e de média confiabilidade para leitura da umidade do solo, mas não restringe que modelos mais sofisticados sejam adquiridos para esta função. Utilizado a comunicação via rádio entre os dois módulos que compõe o projeto, para eliminar o uso de cabos, o que melhora a mobilidade, além da utilização das modernas placa de comunicação WIFI que deixa o agricultor em contato constante com tudo que se passa na sua lavoura através da Internet. A parte do acionamento deixa o projeto amplo, podendo ser utilizado com qualquer tipo de motor, motobomba ou pequenas válvulas solenoides.

O sistema de controle de irrigação baseado na umidade do solo utilizando sistema embarcado cumpre na prática tudo que foi projetado no papel. Aqui apenas um protótipo, mas uma referência de baixo custo para grandes projetos, estruturas de grande porte que pode

atender qualquer tipo de cultivo, qualquer tipo de solo e principalmente qualquer modo de irrigação.

Este trabalho é apenas o início e a base para que outras pesquisas o aperfeiçoe, pois oferece espaço para melhorias que aqui não foram implementadas, mas que possam vir a melhorar o sistema podendo surgir novos modelos de negócio ou apenas torná-lo mais eficientes com a implementação de novos recursos que para o usuário final pode ser de grande valia como por exemplo, medir a fluxo de água para saber o seu consumo, desenvolver um aplicativo para smartphones com acesso *bluetooth*, medir além da umidade a temperatura e acidez do solo que também é muito importante para determinado tipo de plantio, enfim, uma gama de novos recursos que geram novos caminhos de pesquisa e novas oportunidades de melhorias.

Enfim, mostrar que é possível a realização de um trabalho através dos conhecimentos adquiridos e em pesquisas traz uma satisfação impar para que novos produtos e serviços surjam e que torne a vida das pessoas melhores e facilite o seu dia-a-dia de cada um.

REFERÊNCIAS

AGÊNCIA NACIONAL DAS ÁGUAS. **“Atlas Irrigação**: uso da água na Agricultura Irrigada. 2016. Disponível em: http://arquivos.ana.gov.br/imprensa/publicacoes/AtlasIrrigacao-UsodaAguanaAgriculturaIrrigada.pdf. Acesso em: 06/04/2018.

AGÊNCIA NACIONAL DAS ÁGUAS. **GEO Brasil Recursos Hídricos**: Componente da Série de Relatórios sobre o Estado e Perspectivas do Meio Ambiente no Brasil. 2017. Disponível em: http://www.snirh.gov.br/portal/snirh/centrais-de-conteudos/conjuntura-dos-recursos-hidricos/relatorio-conjuntura-2017.pdf. Acesso em: 06/11/2017.

ALBERTS, Patrik. ***Datasheet Logical Level Bidirecional,*** 2013. Disponível em: https://cdn.sparkfun.com/datasheets/BreakoutBoards/Logic_Level_Bidirectional.pdf. Acesso em: 09/09/2018.

ARDUINO E CIA. **Sensor de umidade e temperatura DHT11**, 2016. Disponível em: https://www.arduinoecia.com.br/2013/05/sensor-de-umidade-e-temperatura-dht11.html. Acesso em: 04/05/2018.

ARDUINO.CC. **Arduino *Reference***, 2017. Disponível em: https://www.arduino.cc/reference/en/. Acesso em 20/05/2018.

ARDUINOLANDIA. **Sensor de Umidade de Solo Higrômetro**, 2018.Disponível em: https://www.arduinolandia.com.br/sensor-de-umidade-do-solo-higrometro. Acesso em: 08/05/2018.

ASCÂNIO, Angel. **Energias alternativas:** Diagrama para conectara una bomba de água. 2010. Disponível em: https://angelascanio.wordpress.com/2010/10/06/diagrama-para-conectar-un-panel-solar-a-una-bomba-de-agua/. Acesso em: 20/10/2018.

ATHOS ELECTRONICS. **Contator:** Tipos e dimensionamentos. 2015. Disponível em: https://athoselectronics.com/contator/. Acesso em: 16/06/2018.

BAYLE, Julien. ***C Programming for Arduino***. Birmingham, United Kingdom: Packt Publishing Ltd, 2013.

BEIGHLEY, Lynn. **Use a Cabeça! SQL**. Jacaré, Brasil: Alta Books, 2012.

BERALDO, Roberto. **10 Novidades do PHP 7**, 2015. Disponível em: https://tableless.com.br/10-novidades-do-php-7/. Acesso em: 14/03/2018.

BRASIL. Ministério do Meio Ambiente, Conselho Nacional de Meio Ambiente, CONAMA. **Resolução CONAMA nº 284/2001**, de 30 de Agosto de 2001 – In: Resoluções, 2001. 2016.

Disponível em: http://www2.mma.gov.br/port/conama/legiabre.cfm?codlegi=282. Acesso em: 16/08/2018.

CIRCUITS TODAY. ***Arduino NANO Tutorial, Pinout and Schematics***. 2017. Disponivel em: http://www.circuitstoday.com/arduino-nano-tutorial-pinout-schematics. Acesso em: 24/04/2018.

CONVERGE, Tim; MORGAN Clark; PARK Joyce. ***PHP5 and MySQL, Bible***. Indianápolis, USA: Wiley Publishing, Inc, 2004.

DAVIS, Michele E.; PHILLIPS, Jon A. ***Learning PHP and MySQL***. 2 ed. Sebastopol, USA: O'Reilly Media, Inc, 2007.

DEJAN. **Arduino *Wireless Communication* – nRF24L01 *Tutorial*.** 2017. Disponível em: https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/. Acesso em: 03/10/2018.

DIAS, Fulvio Cesar Canducci. **Pra que serve relacionamento de tabelas?**. 2017.Disponível em: https://social.msdn.microsoft.com/Forums/pt-BR/9f9cce81-ae74-465a-a26e-7585269dabc5/pra-que-serve-relacionamento-de-tabelas?forum=520. Acesso em: 02/04/2018.

DUCKETT, Jon. ***HTML & CSS, Design and Build Websites***. Indianapolis, USA: John Wiley & Sons Inc , 2011.

EIS, Diego; FERREIRA, Elcio. **HTML5, Curso W3C Escritório Brasil**. 2016. Disponível em: http://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf. Acesso em: 05/03/2018.

FAIRCHILD SEMICONDUTOR. **CD4049UBC • CD4050BC Hex *Inverting Buffer* • Hex *Non-Inverting Buffer*.** 2015. Disponível em: https://www.*elektronik-kompendium.de/public/schaerer/FILES*/cd4049_cd4050.pdf.. Acesso em: 08/09/2018.

FAIS, Roni Márcio. **Tipos de Relacionamento em um Banco de Dados**. 2009. Disponível em: http://www.rmfais.com/rmfais/artigos/table.php?_codigo=6. Acesso em: 12/04/2018.

FAO, Organização das Naçoes Unidas para Alimentação e Agricultura. **2050:** A escassez de água em várias partes do mundo ameaça a segurança alimentar e os meios de subsistência. 2015. Disponível em: http://www.fao.org/news/story/pt/item/283446/icode/. Acesso em: 14/04/2018.

FERNANDO K TECNOLOGIA. **Tutoriais, Tecnologia e Tendências.** 2015. Disponível em: https://www.fernandok.com/2017/08/automacao-com-esp8266-utilizando-reles.html. Acesso em: 27/08/2018.

FERREIRA, Valber Mendes. **Irrigação e drenagem.** Floriano, PI: EDUFPI, 2011. p. 15.

FILIPEFLOP. **Válvula de Vazão Solenóide de Água 12VDC**. 2017. Disponível em: https://www.filipeflop.com/produto/valvula-de-vazao-solenoide-agua-12vdc/. Acesso em: 28/04/18.

FRASER, Ben**. Arduino nRF24L01+ *Communications***. 2017. Disponível em: https://medium.com/@benjamindavidfraser/arduino-nrf24l01-*communications*-947e1acb33fb. Acesso em: 01/10/2018.

FRITZEN, Clóvis. **Sensor de umidade do solo**: Calibração e utilização. 2016. Disponível em: https://fritzenlab.com.br/2016/07/sensor-de-umidade-do-solo-calibracao-e-utilizacao/. Acesso em: 20/10/2018.

FRITZING. **16x2 I2C LCD *PART***. 2018. Disponível em: http://forum.fritzing.org/t/16x2-i2c-lcd-part/2041/3. Acesso em: 14/08/2018.

HALVORSEN, Hans Petter. ***Strucutred Query Language***. *University College of Southeast Norway*, USA, 2016.

HARRIS, Andy. ***HTML, XHTML, & CSS All-in-One For Dummies***. 2 ed. Indianapolis, USA: *Wiley Publishing, Inc*, 2011.

HASS, Valdomiro. **Infraestrutura no Campo**. 2017. Disponível em: http://www.agricultura.rs.gov.br/camara-setorial-de-irrigacao-e-infraestrutura. Acesso em: 01/05/2018.

LURIG, Mario. ***PHP Reference: Beginner to Intermediate PHP5***. *Creative Commons Attribution-NonCommercial-ShareAlike*, 2008.

McROBERTS, Michael. **Arduino Básico**. Curitiba, Brasil. Novatec, 2011.

MAKINO. **Aspersores**. 2015. Disponível em: https://makino.com.br/irrigacao-aspersores. Acesso em: 05/05/2018.

MINATEL, Pedro. **Sistemas Embarcados e *Internet* das Coisas.** 2017. Disponível em: http://pedrominatel.com.br/pt/esp8266/esp8266-o-guia-basico-de-*hardware*/. Acesso em: 27/08/2018.

MONK, S. **Programação com Arduino:** começando com *sketches*. Porto Alegre: BOOKMAN EDITORA LTDA, 2013.

MOZILLA. **Como o CSS Funciona.** 2016. Disponível em: https://developer.mozilla.org/pt-BR/docs/Aprender/CSS/Introduction_to_CSS/como_CSS_funciona. Acesso em: 02/03/2018.

MOZILLA. ***JavaScript Basics*.** 2018. Disponível em: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_*web*/JavaScript_basics. Acesso em: 05/03/2018.

NAANDANJAIN. ***A Jain Irrigation Company*.** 2017. Disponível em: https://naandanjain.com.br/irrigacao-automatizada-em-tempos-de-crise-uma-opcao-economica/. Acesso em: 05/10/2018.

NASCIMENTO, Rodrigo. O que é *Dashboard*?. 2017. Disponível em: http://marketingpordados.com/analise-de-dados/o-que-e-dashboard-%F0%9F%93%8A/. Acesso em: 16/09/2014.

NATIONAL INSTRUMENTS. ***What are the Basic Differences Between CMOS and TTL Signals?***. 2016. Disponível em: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000P9yaSAC. Acesso em 21/03/2018.

NEDELKOVSKI, Dejan. ***Control High Voltage Devices – Arduino Relay Tutorial*.** 2017. Disponível em: https://howtomechatronics.com/tutorials/arduino/control-high-voltage-devices-arduino-relay-tutorial/. Acesso em: 10/03/2018.

NIXON, Robin. ***Learning PHP, MySQL, JavaScript, CSS & HTML5***. 4 ed. Sebastopol, USA: O'Reilly Media, Inc, 2015.

NORDBOTTEN, Svein. ***Introduction to PHP5 with MySQL***. Bergen, Noruega: O'Reilly Media, Inc, 2009.

NORDIC SEMICONDUTORES. **nRF24L01:** *Ultra Low Power* 2.4 *G*Hz *RF Transceiver* IC. 2018. Disponível em: https://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01. Acesso em: 13/05/2018.

OLIVEIRA, Viviane. **Irrigação por Gotejamento**. 2015. Disponível em: http://agronegociointerior.com.br/irrigacao-por-gotejamento/. Acesso em: 03/05/2018.

PINHO, Márcio Sarróglia. **Subalgoritmos (Funções)**. 2016. Disponível em: https://www.inf.pucrs.br/~pinho/LaproI/Funcoes/AulaDeFuncoes.htm. Acesso em: 03/04/2018.

REIS, Valdinei Rodrigues. **I2C:** Protocolo de Comunicação. 2015. Disponível em: http://www.arduinobr.com/arduino/i2c-protocolo-de-comunicacao/. Acesso em: 27/04/2018.

RESOURCE SUPPLY, LLC. **IP67:** *What Does That Mean*?. 2008. Disponível em: http://www.resourcesupplyllc.com/PDFs/WhatDoesIP67Mean.pdf. Acesso em: 17/09/2018.

ROCHA, Helder da. **Entradas e Saídas do Arduino**. 2014. Disponível em: http://eletronicaparaartistas.com.br/artigos/. Acesso em: 15/04/2018.

SCHMITZ, Daniel. **Tudo que você queria saber sobre Git e GitHub, mas tinha vergonha de perguntar**. 2015. Disponível em: https://tableless.com.br/tudo-que-voce-queria-saber-sobre-git-e-github-mas-tinha-vergonha-de-perguntar/. Acesso em: 01/04/2018.

SCUDERO, Erick. **5 Aplicações que podem ser criadas aprendendo apenas HTML, CSS e JS**. 2016. Disponível em: https://becode.com.br/aplicacoes-que-podem-ser-criadas-aprendendo-html-css-e-javascript/. Acesso em: 12/04/2018.

SEAAN. **Comandos Elétricos Básicos.** 2014. Disponível em: http://m.seaan.com.br/cursos/comandos-eletricos-basico/apostila/. Acesso em: 09/09/2018.

SILVA, Maurício Samy. **Bootstrap 3.3.5 – Aprenda a usar o framework Bootstrap para criar layouts CSS complexos e responsivos**. São Paulo: Novatec Editora Ltda, 2015.

SILVEIRA, Cristiano Bertulucci. **Tecnologia PWM**. 2013. Disponível em: https://www.citisystems.com.br/pwm/. Acesso em: 22/04/2018.

SOUZA, Fábio. **Arduino *MEGA* 2560**. 2016. Disponível em: https://www.embarcados.com.br/arduino-mega-2560/. Acesso em: 22/04/2018.

SOUZA, Roberto. **Como Dimensionar Contatores Elétricos.** 2014. Disponível em: https://www.robertdicastecnologia.com.br/2014/11/como-dimensionar-contatores-eletricos/. Acesso em: 09/09/2018.

TAWIL, Yahia. ***Understanding Arduino UNO Hardware Design***. 2016. Disponível em: https://www.allaboutcircuits.com/technical-articles/understanding-arduino-uno-hardware-design/. Acesso em: 15/04/2018.

TELECO. **Redes Wi-Fi:** Espectro de Frequência ISM. 2018. Disponível em: http://www.teleco.com.br/tutoriais/tutorialredeswifi1/pagina_5.asp. Acesso em: 14/05/2018.

TESTEZLAF, R. **Irrigação:** Métodos, Sistemas e Aplicações. Campinas: Faculdade de Engenharia Agrícola/UNICAMP 2017.

TOOR TECNOLOGIA. **Servidor e provedor de internet:** Qual a diferença?.2017. Disponível em: http://toor.com.br/blog/2017/03/28/servidor-e-provedor-de-internet-qual-a-diferenca/. Acesso em: 25/10/2018.

VERISIGN, INC. **Como estar online:** O que é um nome de domínio?.2018. Disponível em: https://www.verisign.com/pt_BR/website-presence/online/what-is-a-domain-name/index.xhtml. Acesso em: 27/10/2018.

W3C BRASIL. **Padrões HTML5, CSS3 e Acessibilidade**. 2010. Disponível em: http://www.w3c.br/Cursos/PadroesWebAcessibilidade. Acesso em: 10/03/2018.

W3SCHOOL. **PHP 5 Variables**. 2016. Disponível em: https://www.w3schools.com/php/php_variables.asp. Acesso em: 10/04/2018.

WEG. **Partida e Proteção de Motores**. 2017. Disponível em: http://www.weg.net/catalog/weg/BR/pt/search?text=caw04-31e. Acesso em: 09/09/2018.

WILTON, Paul. McPEAK, Jeremy. ***Beginning JavaScript***. 4 ed. Indianapolis, USA: Wiley Publishing, Inc, 2010.

ANEXO A – Algoritmo do Sistema Central de Controle de Irrigação

```
/*
 * SISTEMA DE MONITORAÇÃO DA UMIDADE DO SOLO
 * TRABALHO DE CONCLUSÃO DE CURSO - ENGENHARIA ELÉTRICA TURMA 2014
 * ALUNO: WILLIAM CRUZ DE OLIVEIRA
 * CRIADO EM: 12/08/2018 POR: WILLIAM CRUZ DE OLIVEIRA.
 * ATUALIZADO EM: 28/10/2018 POR: WILLIAM CRUZ DE OLIVEIRA.
*/

//------------------- nRF24L01+ ----------------------------
#include "nRF24L01.h" //Biblioteca nRF24L01.
#include "RF24.h"     //Biblioteca nRF24L01.
#include "SPI.h"      //Biblioteca de controle MOSI, MISO e ACK do nRF24L01.
RF24 radio(9,10);     //Inicializa o nRF24L01 nos pinos 9(CE) e 10(CSN) do Arduino.
const uint64_t pipe = 0xE6E6E6E6E6E6; //Identificação do canal para um par de nRF24L01.
//-----------------------------------------------------------
//------------------- ESP8266 ESP7 -------------------------
#include <doxygen.h>       //Biblioteca do módulo WiFI ESP8266.
#include <ESP8266.h>       //Biblioteca do módulo WiFI ESP8266.
#define CH_PD 44           //sinal de controle de CH_PD
#define DEBUG true
#define SSID   "ROUTER-CASA"                        //ID da rede WIFI
#define PASSWORD   "123456AbCdEfGhIjKlMnOpQr654321" //Senha da rede WIFI
//-----------------------------------------------------------
//------------------- Servidor Mysql -----------------------
#include <Ethernet.h>
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>
//mac, ip, gateway, máscara do arduino
//byte mac[] = { 0x11, 0x12, 0xBE, 0xEF, 0xFE, 0xED };
byte mac[] = { 0x08, 0xAF, 0x01, 0x00, 0x00, 0x03 };
byte ip[] = { 192, 168, 0, 128 };
byte gateway[] = { 192, 168, 0, 1 };
byte subnet[] = { 255, 255, 255, 0 };
//endereço IP do servidor do banco de dados mysql
byte server_addr[] = { 192, 168, 0, 1 };
//usuário e senha do mysql
char user[] = "root";   //Usuário do banco de dados
char password[] = "";   //Senha do banco de dados

char INSERIR_UMID[] = "INSERT INTO leituras(id_leitura, valor, data, hora) VALUES('%d', '%d', %s, %s)";
char  INSERIR_CONSOL[]  =  "INSERT  INTO  consolidado(id_consolidade,  id_sensor,  id_leitura,  data)
VALUES('%d', '%d', %d, %s)";
char CONSULTA[] = "SELECT nome_sensor, umidminima, umidmaxima FROM sensores WHERE id_sensor = 2";
long head_count = 0;
char BANCODEDADOS[] = "USE irrigacaodb";
//-----------------------------------------------------------
//--------------------- DISPLAY LCD -------------------------
```

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
//i2c pins
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
//-----------------------------------------------------------
//----------------------- OUTROS ---------------------------
#include <NTPClient.h> //Atualização de data e hora
#include <WiFiUdp.h>

// Variáveis para salvar data e hora
String formattedDate;
String dayStamp;
String timeStamp;
Int Cont = 0;

// Define NTP Client to get time
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);

//Definições do servidor Arduino WEB
EthernetServer server(80); //porta
float UMID = 0;        //Valor da umidade.
int ValorRecebido = 0; //Valor recebido do sistema monitor de umidade.
int ValorPorcento = 0;   //Valor da umidade em porcentagem.

//Valores lidos do banco de dados
char sentenca[128];
char sentenca1[128];
char nome_sensor[];
int umidminima = 0;
int umidmaxima = 0;

//Definição das portas do relé
int porta_rele1 = 30;
int porta_rele2 = 31;
int porta_rele3 = 32;
int porta_rele4 = 33;
int porta_rele5 = 34;
int porta_rele6 = 35;
int porta_rele7 = 36;
int porta_rele8 = 37;

char[] Modo = "Local";
//-----------------------------------------------------------

void setup()
{
  //Define pinos para o rele como saida
  void inicializa_reles()
```

```
lcd.begin(16,2); //Define o tipo de display 16x2;
lcd.backlight(); //Acende a luz de Backligth do Display

// inicializa a comunicacao serial
Serial.begin(9600);
Serial1.begin(115200);

// Rádio nR24L01+ ----------------------------------------------
radio.begin();                //Inicializa o NRF24L01
radio.openWritingPipe(pipe);  //Inicializa o canal definido em Pipe.
radio.startListening();       //Radio pronto para transmissão.
// ------------------------------------------------------------

// Inicialização WIFI ------------------------------------------
pinMode(CH_PD,OUTPUT);
digitalWrite(CH_PD,HIGH); //Setado em alto - funcionamento normal
delay (1000);

lcd.setCursor(0,0); //Posiciona na primeira linha e primeira coluna do Display
lcd.print("CONECTANDO ...");
lcd.setCursor(0,1);
lcd.print("REDE WIFI...");
delay(2000);

lcd.setCursor(0,0); //Posiciona na primeira linha e primeira coluna do Display
lcd.print("DEFININDO MODO");
lcd.setCursor(0,1);
lcd.print("DE OPERAÇÃO ...");
sendData("AT+CWMODE=1rn", 500, DEBUG); // Modo Estação (cliente)
delay(2000);

lcd.setCursor(0,0); //Posiciona na primeira linha e primeira coluna do Display
lcd.print("USUARIO E SENHA");
lcd.setCursor(0,1);
lcd.print("DA REDE WIFI...");
String  cmd = "AT+CWJAP=\"";
  cmd += SSID;
  cmd += "\",\"";
  cmd += PASSWORD;
  cmd += "\"";
sendData(cmd, 2000, DEBUG);
delay(5000);

while (ESP8266.status() != WL_CONNECTED) || (Cont <= 240) {
  delay(500);
  lcd.setCursor(0,0);
  lcd.print("AGUARDANDO ...");
  lcd.setCursor(0,1);
  lcd.print("REDE WIFI");
  Cont++;
```

```
}

if (Cont == 240)
{
  lcd.setCursor(0,0);
  lcd.print("FALHA CONECTAR!");
  lcd.setCursor(0,1);
  lcd.print("WIFI NÃO DISP.");
  delay(2000);
  lcd.print("MODO OPERACAO:");
  lcd.setCursor(0,1);
  lcd.print("LOCAL.");
  delay(500);
  Modo = "Local";
}else
{
  lcd.setCursor(0,0);
  lcd.print("WIFI CONECTADO!");
  Modo= "Remoto";
  sendData("AT+CIFSRrn", 500, DEBUG); // rst
  delay(1000);
}

//Ajuste da Data e Hora
timeClient.begin();
// Setando o offset da hora para segundos e definindo o TimeZone
// GMT -1 = -3600
// GMT -2 = -7200
// GMT -3 = -10800
// GMT 0 = 0
timeClient.setTimeOffset(-10800);

 //Inicializa servidor html arduino
server.begin();

if (conn.connect(server_addr, 3306, user, password) && Modo != "Local")
{
   delay(200);
   //Conecta com o banco de dados Mysql
   MySQL_Cursor *cur_mem = new MySQL_Cursor(&conn);
   cur_mem->execute(BANCODEDADOS);
   delete cur_mem;

   conn.cmd_query(QUERY_POP);
   conn.get_columns();
   row_values *row = NULL;
   do
   {
      row = conn.get_next_row();
      if (row != NULL)
```

```
        {
          nome_sensor = atol(row->values[0]);
          umidminima = atol(row->values[1]);
          umidmaxima = atol(row->values[2]);
        }
        conn.free_row_buffer();
     } while (row != NULL);
     conn.free_columns_buffer();
  }
  else
  {
     lcd.setCursor(0,0);
     lcd.print("FALHA CONECTAR!");
     lcd.setCursor(0,1);
     lcd.print("MODO LOCAL");
     delay(2000);
     conn.close();
  }
  // ----------------------------------------------------------------
}

void loop()
{
  formattedDate = timeClient.getFormattedDate();

  int splitT = formattedDate.indexOf("T"); // Extraindo a data
  dayStamp = formattedDate.substring(0, splitT);
  timeStamp = formattedDate.substring(splitT+1, formattedDate.length()-1); // Extraindo a hora

  radio.read(ValorRecebido, 1); // Valor recebido do rádio.
  //ValorPorcento = map(sensorValue, 0, 1023, 0, 100); // Converte o valor recebido em porcentagem

  lcd.setCursor(0,0);
  lcd.print("SENSOR: %s",nome_sensor);
  lcd.setCursor(0,1);
  lcd.print("VALOR: %s",ValorRecebido);

 if (ValorPorcento > -1)
 {
   sprintf(sentenca, INSERIR_UMID[], ValorPorcento, dayStamp, timeStamp); //Monta a query e salva na
variável sentenca
   sprintf(sentenca1, INSERIR_CONSOL[], id_sensor, id_leitura, dataStamp); //Monta a query e salva na
variável sentenca1

   MySQL_Cursor *cur_mem = new MySQL_Cursor(&conn);     //Conecta ao banco de dados
   cur_mem->execute(sentenca);                          //Executa a query
   delete cur_mem;                                      //Limpa a memória
   cur_mem1->execute(sentenca1);                        //Executa a query
   delete cur_mem1;                                     //Limpa a memória ?
```

```
  if (ValorRecebido <= umidminima)
  {
    digitalWrite(porta_rele1, LOW); //liga rele 1
    delay(500);
  }
  else if (ValorRecebido >= umid maxima)
  {
    digitalWrite(porta_rele1, HIGH); //desliga rele 1
    delay(500);
  }
 }
 else
 {
   sprintf(sentenca, INSERIR_UMID[], "nan", "nan", "nan"); //Monta a query e salva na variável sentenca
   sprintf(sentenca1, INSERIR_CONSOL[], 0, 0, "nan"); //Monta a query e salva na variável sentenca1

   MySQL_Cursor *cur_mem = new MySQL_Cursor(&conn);    //Conecta ao banco de dados
   cur_mem->execute(sentenca);                         //Executa a query
   delete cur_mem;                                     //Limpa a memória ?
 }
}

void inicializa_reles()
{
  pinMode(porta_rele1, OUTPUT);
 digitalWrite(porta_rele1, HIGH);
 pinMode(porta_rele2, OUTPUT);
 digitalWrite(porta_rele2, HIGH);
 pinMode(porta_rele3, OUTPUT);
 digitalWrite(porta_rele3, HIGH);
 pinMode(porta_rele4, OUTPUT);
 digitalWrite(porta_rele4, HIGH);
 pinMode(porta_rele5, OUTPUT);
 digitalWrite(porta_rele5, HIGH);
 pinMode(porta_rele6, OUTPUT);
 digitalWrite(porta_rele6, HIGH);
 pinMode(porta_rele7, OUTPUT);
 digitalWrite(porta_rele7, HIGH);
 pinMode(porta_rele8, OUTPUT);
 digitalWrite(porta_rele8, HIGH);
}
```

ANEXO B – Algoritmo do Sistema Monitor de Umidade

```
/*
 * SISTEMA DE MONITORAÇÃO DA UMIDADE DO SOLO
 * TRABALHO DE CONCLUSÃO DE CURSO - ENGENHARIA ELÉTRICA TURMA 2014
 * ALUNO: WILLIAM CRUZ DE OLIVEIRA
 * CRIADO EM: 14/08/2018 POR: WILLIAM CRUZ DE OLIVEIRA.
 * ATUALIZADO EM: 28/10/2018 POR: WILLIAM CRUZ DE OLIVEIRA.
*/

#include "nRF24L01.h" //Biblioteca nRF24L01.
#include "RF24.h"     //Biblioteca nRF24L01.
#include "SPI.h"      //Biblioteca de controle MOSI, MISO e ACK do nRF24L01.
#include “Time.h”     //Hora do Sistema

RF24 radio(9,10);     //Inicializa o nRF24L01 nos pinos 9(CE) e 10(CSN) do Arduino
const uint64_t pipe = 0xE6E6E6E6E6E6; //Identificação do canal para um par de nRF24L01.

//Constantes do algorítmo:
const int SensorPin = A0;   // Entrada analogica do Sensor de Umidade

int leituraSensor = 0;      // leitura do sensor de umidade.
int leituraPorCento = 0;    // valor em porcentagem para enviar ao Rádio receptor.
int vminimo = 395;          // Valor mínimo com o máximo de umidade
int vmaximo = 1023;         // Valor máximo lido com umidade no mínimo

void setup()
{
  Serial.begin(9600); // inicializa a comunicacao serial.
  pinMode(SensorPin, INPUT);    //Define a porta de leitura do sensor de umidade.
  radio.begin();                //Inicializa o NRF24L01
  radio.openWritingPipe(pipe);  //Inicializa o canal definido em Pipe.
  radio.startListening();       //Radio pronto para transmissão.
}

void loop()
{
  formattedDate = timeClient.getFormattedDate();
  int splitT = formattedDate.indexOf("T"); // Extraindo a data
  timeStamp = formattedDate.substring(splitT+1, formattedDate.length()-1); // Extraindo a hora

  if ((timeStamp == "00:00:00") || (timeStamp == "00:30:00") || (timeStamp == "01:00:00") || (timeStamp
== "01:30:00") || (timeStamp == "02:00:00") || (timeStamp == "02:30:00") || (timeStamp == "03:00:00") ||
(timeStamp == "03:30:00") || (timeStamp == "04:00:00") || (timeStamp == "04:30:00") || (timeStamp ==
"05:00:00") || (timeStamp == "05:30:00") || (timeStamp == "06:00:00") || (timeStamp == "06:30:00") ||
(timeStamp == "07:00:00") || (timeStamp == "07:30:00") || (timeStamp == "08:00:00") || (timeStamp ==
"08:30:00") || (timeStamp == "09:00:00") || (timeStamp == "09:30:00") || (timeStamp == "10:00:00") ||
(timeStamp == "10:30:00") || (timeStamp == "11:00:00") || (timeStamp == "11:30:00") || (timeStamp ==
"12:00:00") || (timeStamp == "12:30:00") || (timeStamp == "13:00:00") || (timeStamp == "13:30:00") ||
```

```
(timeStamp == "14:00:00") || (timeStamp == "14:30:00") || (timeStamp == "15:00:00") || (timeStamp ==
"15:30:00") || (timeStamp == "16:00:00") || (timeStamp == "16:30:00") || (timeStamp == "17:00:00") ||
(timeStamp == "17:30:00") || (timeStamp == "18:00:00") || (timeStamp == "18:30:00") || (timeStamp ==
"19:00:00") || (timeStamp == "19:30:00") || (timeStamp == "20:00:00") || (timeStamp == "20:30:00") ||
(timeStamp == "21:00:00") || (timeStamp == "21:30:00") || (timeStamp == "22:00:00") || (timeStamp ==
"22:30:00") || (timeStamp == "23:00:00") || (timeStamp == "23:30:00"))
{
        leituraSensor = analogRead(analogInPin);  //faz a leitura do sensor de umidade;
        //Transforma a leitura do sensor em porcentagem
        leituraPorCento = map(leituraSensor, vminimo, vmaximo, 0, 100);
        radio.write(leituraPorCento,1); //Transmite o valor lido para o outro rádio.
  }
}
```