CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ – UNIDAVI

GABRIEL FIAMONCINI

PROTÓTIPO DE AUTOMAÇÃO DE BAIXO CUSTO PARA PEQUENAS ÁREAS DE CULTIVO PROTEGIDO

RIO DO SUL

2021

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ – UNIDAVI

GABRIEL FIAMONCINI

PROTÓTIPO DE AUTOMAÇÃO DE BAIXO CUSTO PARA PEQUENAS ÁREAS DE CULTIVO PROTEGIDO

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí, como condição parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Orientador: Marcondes Maçaneiro

RIO DO SUL

2021

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ – UNIDAVI

GABRIEL FIAMONCINI

PROTÓTIPO DE AUTOMAÇÃO DE BAIXO CUSTO PARA PEQUENAS ÁREAS DE CULTIVO PROTEGIDO

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí- UNIDAVI, a ser apreciado pela Banca Examinadora, formada por:

Professor Orientador: Marcondes Maçaneiro

Banca Examinadora:

Prof. Fernando Andrade Bastos

Prof. Jeancarlo Visentainer

RIO DO SUL

2021

AGRADECIMENTOS

Agradeço aos meus pais Odair e Maria, por sempre acreditarem em mim, além de sempre me incentivarem a continuar estudando, agradeço ao meu irmão Leonardo por sempre me passar a sua determinação de não desistir e a minha namorada Astrid por ser compreensiva e sempre me incentivar. Agradeço aos professores pela paciência e por passar seus conhecimentos sempre da melhor maneira possível.

RESUMO

Com o passar dos anos os produtores passam por instabilidade no processo do cultivo em um contexto geral, sendo um dos principais fatores que levam a esse problema, a constante alteração climática, horas severas, outrora brandas. Os Produtores de pequeno e médio porte neste cenário, buscam maneiras de melhorar a sua qualidade de produção, por meio disso, a cada dia, estão mais integrados dos avanços da tecnologia para a agricultura e cultivo. Constantemente temos avanços com forma de cultivo, sendo menos evasivas e agressivas ao ambiente, porém muitas vezes fora do alcance financeiro. Este trabalho tem como objetivo, desenvolver um protótipo de uma estufa automatizada, visando criar um ambiente controlado e monitorado onde o usuário tenha todas as informações de fácil acesso, sincronizadas, além de um valor atrativo para investimento. O desenvolvimento desse protótipo de estufa, consistem em utilizar hardwares de baixo custo de aquisição, sendo eles o microcontrolador Arduino, como coletor de dados e o microcomputador Raspberry Pi, atuando como servidor de dados. Para a conversação entre os hardwares, foi utilizado a linguagem de programação C, a qual se encarrega de fazer a leitura dos sensores, para o processamento das informações e armazenagem foi utilizado o banco de dados Mysql, no microcomputador Raspberry Pi fica alocado o painel de informações, foi utilizado a linguagem PHP para a criação de uma dashboard com as informações coletadas. Com o desenvolvimento do protótipo foi identificado que há como implementar melhorias uma futura atualização do protótipo, possibilitando uma versão mais afinada.

Palavras-Chave: Arduino, Raspberry Pi, Automação.

ABSTRACT

Over the years, producers experience instability in the cultivation process in a general context, one of the main factors that lead to this problem, the constant climate change, severe hours, which were once mild. Small and medium-sized producers in this scenario are looking for ways to improve their production quality, through which, every day, they are more integrated with advances in technology for agriculture and cultivation. We constantly have advances in the form of cultivation, being less evasive and aggressive to the environment, but often out of financial reach. This work aims to develop a prototype of an automated greenhouse, aiming to create a controlled and monitored environment where the user has all information easily accessible, synchronized, in addition to an attractive investment value. The development of this greenhouse prototype consists of using low-cost hardware, such as the Arduino microcontroller, as a data collector, and the Raspberry Pi microcomputer, acting as a data server. For the conversation between the hardware, the C programming language was used, which oversees reading the sensors, for the processing of information and storage, the MySQL database was used. PHP language was used to create a dashboard with the collected information. With the development of the prototype, it was identified that there is a way to implement improvements in a future version, enabling a more refined version of the prototype.

Keywords: Arduino, Raspberry Pi, Automation.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface.

ARM – Advanced RISC Machine

EEPROM – Electrically-Erasable Programmable Read-Only Memory.

EXE – Executable File.

GPIO – General Purpose Input / Output

HTML – HyperText Markup Language.

I2C – Inter-Integrated Circuit.

IMAP – Internet Message Access Protocol.

LCD – Liquid Crystal Display.

LDAP – Lightweight Directory Access Protocol.

Open Source – Software livre de licenças.

POP3 – Post Office Protocol.

QUERY – Consulta

RAM – Random Access Memory.

SGBD – Data Base Management System.

SPI – Serial Peripheral Interface.

SQL – Structured Query Language.

SRAM – Static random-access memory.

USART – Universal Synchronous Asynchronous Receiver Transmitter.

USB – Universal Serial Bus.

UTF8 – 8-bit Unicode Transformation Format

VIEW – Virtual Tables

SUMÁRIO

1. INTRODUÇÃO

Nos conta Soglio (2008) que atualmente produtores de pequeno e grande porte sofrem constantemente com perdas na produção em lavouras em campo aberto tendo-se como o principal fator à contínua variação climática, sendo cada vez mais severa ao decorrer dos anos.

Além da contribuição por outros fatores importantes, como a existência de pragas e doenças na área do cultivo. Buscando diminuir a incidência desses fatores, especialmente para produtos de origem alimentícia, têm-se buscado soluções com o auxílio da tecnologia, aliada a automação para à construção de ambientes para plantio e cultivo protegidos os quais podem ser monitorados e controlados.

No atual cenário descreve Petry *et al*. (2008) que possuímos um arsenal de ferramentas e maneiras, onde há estruturas que atendem pontualmente as necessidades agrícolas, ou seja, são específicos para irrigação, ou controle de temperatura e umidade, ou simplesmente controle interno da ventilação e distribuição de fertilizantes. A aparente complexidade de especificar a necessidade, bem como a falta de conhecimento tácito, induz a procura de empresas especializadas, onde normalmente é oferecido uma solução completa, que por sua vez, além de possuir etapas para a implementação, gera um alto investimento, fator que exclui uma grande parcela dos agricultores.

Centros Acadêmicos de Tecnologia em parceria com investidores interessados, estudam maneiras sustentáveis e com baixo custo para proporcionar uma solução que atenda não só pontualmente, mas que possa ser flexível à necessidade do produtor, buscando proporcionar um ambiente onde possa ser controlado e monitorado, além de ser de fácil manipulação.

Com base nas informações supracitadas acima este projeto visa descrever a construção de um protótipo de uma estufa controlada. O projeto foi constituído com sensores para medição e controle da temperatura e umidade, controle de ventilação e iluminação. O projeto também busca ser escalável para a integração de muitos componentes, além de ser adaptável a área disponível. Por fim, procurou-se utilizar componentes de baixo custo disponíveis no mercado atual.

No auxílio das referências bibliográficas, traz-se uma definição teórica e histórica das ferramentas, linguagens de programação e um apanhado geral do hardware que foram utilizados nesse protótipo, bem como estruturar como será construído. Ademais o propósito deste trabalho é demonstrar um protótipo de uma estufa escalável, podendo assumir diferentes configurações de ajustes ao tipo de cultura desejado, além de possuir uma interface de fácil operação ao usuário, bem como uma fácil implantação, com custo baixo ao produtor.

1.1 PROBLEMA DE PESQUISA

Como um protótipo de automação de baixo custo pode auxiliar no cultivo protegido onde há muita variação climática e pouco espaço?

1.2 OBJETIVOS

1.2.1 Geral

- Apresentar um protótipo de automação para estufa de pequena escala, utilizando-se de componentes de baixo custo para cultivo protegido.

1.2.2 Específicos

- Aplicar tecnologias de baixo custo como Arduino e Raspberry Pi para desenvolvimento do protótipo.
- Desenvolver o programa para leitura e envio de dados entre os dispositivos utilizados.
- Criar um protótipo de interface para controle e monitoramento.

1.3 JUSTIFICATIVA

O cenário da agricultura tem sofrido com as alterações climáticas cada vez mais severas com o passar do tempo, muitas vezes inviabilizando, parcial ou totalmente a área de cultivo. Leva-se em questão que no mercado agrícola atual.

A tecnologia vem sendo cada vez mais facilitada para esse público, antes a aquisição de equipamentos para tratar de controle e monitoria só era acessível a preço elevado. Estudos nos confirmam que esta área da agricultura em geral, está em constante crescimento tecnológico, visto que é uma boa área para trabalho no foco da automação dos processos com uso de tecnologia embarcada e de baixo custo.

Com a distribuição desse segmento embarcado esses efeitos vêm sendo amenizados, hoje um produtor rural com baixo investimento, pode possuir em sua residência um espaço dedicado, no qual pode adquirir equipamentos para especificar padrões para o ambiente, controlando os dados de uma determinada cultura. Além da possibilidade se segregar esse ambiente para dois ou mais tipos de produção. Além do custo ser baixo para a implementação de automação, o projeto pode ser escalável, ou até mesmo replicado para outra área.

Por este motivo produtores e tecnologia, cada vez caminham mais próximas. A aplicação de automação nesse contexto visa praticidade e auxílio, deixando de forma mais branda rotinas antes realizadas manualmente ou por sua vez semiautomáticas, buscando resolver problemas com percas e melhorando resultados produtivos.

2. REFERENCIAL TEÓRICO

Neste tópico do trabalho será abordado os temas bases para a estrutura do desenvolvimento do protótipo de automação. Foi dividido em tópicos, onde mostra uma introdução sobre a linguagem PHP e C, responsáveis por realizar a parte de integração dos serviços. Para a questão de armazenamento de dados, há uma descrição do banco de dados Mysql, o qual será responsável por manter os dados íntegros e disponíveis para consumo.

Nas questões de hardware, há uma breve introdução sobre o microcontrolador Arduino, que se responsável por coletar os dados. Finalizando esse tópico, há uma abordagem sobre o microprocessador Raspberry Pi, de suma importância para o protótipo, onde ele será encarregado de integrar todos os dados.

Software é usado para designar um emaranhado de programas, os quais são utilizados para um sistema. (WEBER, 2008, p.256) fala que, “um programa é uma lista de instruções ou comando que determina ao computador a execução de uma tarefa de processamento de dados.”. Além de que atualmente possuímos diversas linguagens de programação na, das quais, utilizamos para desenvolver aplicações para as mais variadas plataformas.

Detalha Weber (2008) que, os programas de computadores atualmente são escritos em linguagens de alto nível, nada mais é, que um alto nível de abstração, não sendo necessário lidar diretamente a níveis de máquina. Basicamente um software ou programa, consiste em seguir uma especificação, sendo ela direta ou indireta, de má sequência de instruções de máquina, como abrir ou fechar determinada porta. Linguagens de programação como referencial geral ao estudo da estrutura das diversas linguagens de programação de alto nível, tratadas separadamente do hardware.

> Um programa de computador é uma lista de instruções ou comandos que determina ao computador a execução de uma tarefa de processamento e dados. Há diversas linguagens de programação, através das quais se podem escrever programas para o computador, mas o computador só pode executá-los quanto eles estão representados internamente de forma binária. (WEBER,2008, p.255).

Para Weber (2008) os programas que não estão escritos de forma binária necessitam ser traduzidos antes de serem executados, os quais podem ser incluídos em categorias, detalhadas no Quadro 1.

Quadro 1 – Categoria de códigos traduzidos

Código binário	Sequência de instruções e operações em binário, representação exata de como as instruções estão na memória.
Código octal ou hexadecimal	Representa de forma idêntica o código binário, mudando apenas a apresentação.
Código Simbólico	O desenvolvedor atribui símbolos, deles quais, letras, números além de caracteres especiais, são utilizados para representar o código de instruções.
Linguagens Alto Nível	Alto nível de abstração, não sendo necessário o desenvolvedor trabalha diretamente com as operações, podendo atribuir funções para o hardware, porém sem tratar diretamente.

Fonte: Weber (2008, p.258)

2.1 - LINGUAGEM DE PROGRAMAÇÃO PHP

O PHP foi orginalmente chamado de Personal Home Page Tools, atualmente desenvolvedores ainda acreditam ser esse o significado do acrônimo, PHP por sua vez é uma linguagem de criação de scripts do lado servidor, que pode ser incorporada em HTML ou utilizada como um binário independente, segundo Converse e Park (2003).

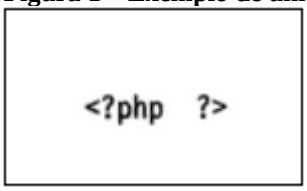
> Rasmus Lerdorf, engenheiro de software, membro da equipe Apache e o homem internacional do mistério é o criador e a força propulsora original por trás do PHP. A primeira parte do PHP foi desenvolvida para sua utilização pessoal no final de 1994.Tratava-se de um wrapper(“envoltório”) de CGI que o auxiliava a monitorar as pessoas que acessavam seu site pessoal. (CONVERSE; PARK, 2003, p.4).

Converse e Park (2003), definem que o PHP é uma linguagem de programação real, diferente de outras da época, como a citada ColdFusion, PHP é construído por um conjunto de TAGs predefinidas, similar ao HTML, no entanto, o desenvolvedor pode definir funções da forma que necessitar aos seus códigos simplesmente digitando um nome e uma definição, conforme exemplificado.

Em questão de estabilidade Converse e Park (2003, p.10), definem que, “entre os mais importantes servidores web, o servidor Apache é considerado o mais estável, com uma reputação invejável com relação aos percentuais de tempo de funcionamento.” Por sua vez o PHP herda essa confiabilidade e sua implementação se torna sólida.

A flexibilidade que o PHP proporciona o faz ser compatível com os mais populares bancos de dados, além de sua base suportar protocolos mais utilizados e importantes na internet, como POP3, IMAP e LDAP, como descrevem Converse e Park (2003). Basicamente um programa em PHP como informa Ramos, et al (2007) é o resultado de uma coleção de super TAGs de HTML. A arquivo é identificado como PHP (extensão .php), deve ser disponibilizado por um servidor Web ativo e conectado a um interpretador PHP, no qual, o interpretador ignora todo e qualquer texto e passa a identificar somente o conteúdo entre as TAGs conforme demonstrado na Figura 1.

Figura 1 – Exemplo de uma TAG PHP



Fonte: Converse (2008, p.46)

Niederauer (2008), complementa que uma página PHP não contém apenas códigos de programação PHP, mas também em seu escopo, possui TAGs de marcação HTML, por sua vez o PHP representa na página a parte não visível aos usuários enquanto o HTML trata da estética, permitindo que em uma estrutura produzida em HTML possua encapsulado códigos em PHP, deixando o projeto dinâmico e transparente ao usuário final.

Savoia (2013), descreve que o PHP foi criado para possuirmos sites dinâmicos, principal vantagem comparando com as linguagens da época. A simplicidade está no PHP por ser de fato uma linguagem interpretada, diferente do processo de linguagens compiladas. “Interpretada porque o PHP trabalha direto com o código da fonte ao ser executado, diferente das linguagens compiladas, que necessitam de um arquivo binário, por exemplo, os .EXE.” (Savoia,2013, p.16). Explicando um pouco como a linguagem PHP é interpretada e processada, precisamos contar com um servidor Web que se encarregue do processo, por sua vez, no pacote de instalação disponibilizado do PHP, encontramos o servidor Apache comenta Savoia (2013).

> O servidor Apache foi criado em 1995 por Rob McCool e presenta a principal tecnologia da Apache Software Foundation, empresa responsável por vários projetos que envolvem tecnologias de transmissão via Web, processamento de dados e execução de aplicativos distribuídos. (SAVOIA,2013, p.11).

Informa Savoia (2013) que com a instalação das ferramentas necessárias para o desenvolvimento em PHP, por ser excelente e criar páginas dinâmicas, não tem capacidade de gerenciar e armazenar os dados por ele processados, pois necessita de uma ferramenta para manipulação dessas informações, aplicando o uso de banco de dados. “O termo banco de dados foi criado inicialmente pela comunidade computacional para indicar coleções organizadas de dados armazenados em computadores digitais.”. (RAMOS,2007, p.12).

Responsáveis por manipular as informações coletadas, Ramos (2007) descreve que para gerenciar essas informações precisamos de um SGBD popularmente conhecidos como sistemas gerenciadores de banco de dados. O SGBD por sua vez, tem como função principal, facilitar a maneira de como os dados coletados serão armazenados, além de proporcionar formas de acesso e integração com o desenvolvedor.

Colocando em uma linguagem mais técnica, Ramos (2007), informa que um banco de dados é uma concentração de informações que são salvos em um computador de modo sistemático, da qual outro computador, possa acessar essas informações de forma prática e rápida sendo muito transparente ao desenvolvedor. “E com certa frequência o banco de dados Mysql é escolhido pelos desenvolvedores PHP.”. (SAVOIA, 2013, p.15).

> MySQL funciona maravilhosamente bem com o PHP, já que consegue arquivar e gerenciar altos níveis de informação com rapidez. E a grande vantagem de se usar PHP e MySQL é que as ferramentas rodam em diversos sistemas operacionais como o Linux, Mac OS e Windows. (SAVOIA,2013, p.15).

Savoia (2013) define que a relação entre PHP e MySQL funciona muito bem, pois ambos foram desenvolvidos um para o outro, em outras palavras, são estruturados para terem melhor desempenho quando trabalhando juntos, além de serem gratuitos (Open Source), recebem melhorias contínuas das comunidades de software.

2.2 – BANCO DE DADOS MYSQL

Um banco de dados trata-se de um espaço para guardar as informações para consulta, alteração ou qualquer outra manipulação quando necessário pelo desenvolvedor ou solicitado pelo próprio usuário final da aplicação, relata Carvalho (2015). Neste contexto ainda comenta que normalmente o modelo que o banco de dados utilizado como muita frequência é o modelo de dados relacional, dentre os demais. “[...], facilidade de alteração da estrutura das tabelas, como adicionar e excluir colunas e linhas de acordo com as necessidades, sem comprometer sua funcionalidade.”. (CARVALHO, 2015, p.11).

> Independentemente do aplicativo que se deseja usar para o armazenamento e manipulação das informações, todos os bancos de dados são construídos por elementos básicos: campos, colunas, linhas ou tuplas e tabelas. Campos são os espaços reservados para inserção de um determinado dados; as colunas são os registros de inserção de um determinado dados; as colunas são os registros de um determinado campo; as tuplas são as linhas de registros de um conjunto de campos; e as tabelas são os conjuntos de linhas, campos e colunas. (CARVALHO, 2015, p.13).

Aponta Milani (2007) que banco de dados MySQL é um servidor de dados bem como um gerenciador de dados relacional, criado para uso em aplicações de pequeno e grande porte, em suas características internas, sustenta todas as ferramentas de outras soluções de banco de dados com grande porte, apontado por muitos utilizadores como a solução de manipulação de dados Open Source, com maior capacidade para confrontar programas similares e de código fechado.

Contando um pouco da história, Milani (2007), ilustra que o MySQL originou-se na década de 90, os desenvolvedores responsáveis por ele foram David AxMark, Allan Larsson e Michael Widerniuns, os quais, necessitavam de uma interface em SQL compatíveis com as rotinas utilizadas em seus sistemas na época, contudo o que tinha disponível não era rápido o suficiente, depois de muito estudo utilizando uma API intitulada mSQL, em linguagem de programação C e C++, criaram uma nova API que deu origem ao MySQL. “Com ótimo resultado gerado por essa nota API o MySQL começou a ser difundido e seus criadores fundaram a empresa responsável por sua manutenção, que é a MySQLAB.” (MILANI,2007, p.23).

> O MySQL, além de banco de dados, contém todas as características de um SGBD, que é o MySQLServer. Além de armazenar os dados, a ferramenta provê todas as características de multiacesso a estes. Entre outras funcionalidades de um SGBD, como, por exemplo, gerenciamento de acesso, integridade dos dados e relacional, concorrência, transações, entre outros. (MILANI,2007, p.26).

Em questão de performance, Milani (2007) informa que o MySQL é rápido em suas transações, pelo fato do SQL utilizado em suas funções ser altamente otimizado pelos desenvolvedores, todavia maioria das novas ferramentas do SQL não são adaptadas ao MySQL pelo fato de poder comprometer o desempenho das já existentes. Ramos, Silva, Álvaro e Afonso (2007, p. 29), listam as características mais importantes e presentes no MySQL, dentre as quais, demonstra algumas características além de um comparativo retratadas na Figura 1, demonstrando a performance em questões de gravação de dados e leitura de dados. O autor aponta ainda algumas características como:

- Portabilidade (suporta praticamente qualquer plataforma atual);
- Compatibilidade (várias formas de conexão, por possuir múltiplos drives para diversas linguagens);
- Excelente desempenho e estabilidade;
- Exige pouco recurso de hardware;
- Uso simplificado (não é complexo de se iniciar o estudo);
- Suporte a diversos padrões de tabelas.

Quadro 2 - Comparativo de Performance entre bancos de dados

Banco de dados – Leitura de 2.000.000 por índice	Segundos
mysql_odbc	464
db2_odbc	1,206
ms-sql-odbc	1,634
Oracle-odbc	20.800
sysbase_odbc	17.614
Banco de dados – Inserção de 350.768 linhas	Segundos
mysql_odbc	619
db2_odbc	3.460
ms-sql-odbc	4.012
Oracle-odbc	11.291
sysbase_odbc	4.802

Fonte: Ramos (2007, p.31)

Podemos ter como ainda complementar algumas palavras sobre MySQL “[...], os serviços Web de acesso a dados utilizam a ‘dobradinha’ Apache/MySQL como forma de minimizar substancialmente o custo de manter uma estrutura de armazenamento de dados.”. (RAMOS, et al.,2007, p.16).

2.3 – LINGUAGEM DE PROGRAMAÇÃO C

Descreve Junior (2019), que em meados da década de 1970, no começo das primeiras versões do sistema operacional UNIX, os desenvolvedores da época encontraram dificuldades, das quais promoveu o estudo para a criação de uma nova linguagem de programação, a fim de minimizar a escrita de programas em linguagem de máquina.

> Briam Kernighan e Dennis Ritcher, que participavam desse projeto, criaram e implementaram uma nova linguagem de programação com todas essas características e, devido à influência de uma outra linguagem denominada B, resolveram batizá-la simplesmente de C. A linguagem de programação C, a partir deste ponto, tornou-se parte integrante dos sistemas operacionais UNIX, compartilhando uma parte de seu sucesso. (JANDL JUNIOR,2019, p.13).

Relatando sobre os fatos do início da linguagem de programação C, Junior (2019) comenta que a linguagem teve uma rápida popularização em razão de sua sintaxe ser de simples entendimento, além de bastante reduzida em comandos, comparando com programas escritos em linguagem de máquina conhecida como Assembly.

Comenta Cocain (2004), que uma das grandes vantagens da linguagem C, está por ela ser de “alto nível” e de “baixo nível” ao mesmo tempo, neste ponto, detalhe que esse nível misto de abstração, se refere ao fato de permitir o controle total da máquina, conjunto de hardware e software, por parte do desenvolvedor, onde permite que seja efetuadas ações sem depender somente do sistema operacional.

> A linguagem C é frequentemente denominada de linguagem de “nível médio”. Isso não ´no sentido de capacidade de processamento entre as linguagens de alto e as de “baixo nível”, mas sim no sentido da sua capacidade de acessar as funções de “baixo nível”, e ao mesmo tempo, de construir os blocos de construção para constituir uma linguagem de “alto nível”. (COCAIN,2004, p.31).

Dentre os tópicos abordados, Cocain (2004), lista as características da linguagem C, abordando em tópicos:

- Linguagem de “alto nível”, sintaxe estruturada.
- Programas escritos em C, são compilados gerando executáveis.
- Compartilha de recursos de alto e baixo nível, onde permite programação direta ao hardware.
- Estruturalmente simples e portável.

Portabilidade é como Junior (2019), descreve a linguagem C, detalha como um programa já desenvolvido pode ser modificado e reescrito para outro sistema distinto do qual foi originalmente criado, pois antigamente era comum um mesmo programa ser escrito na mesma linguagem, porém com variações internas, de maneira que fosse atender ao requisito de outro sistema.

> Um esforço significativo para a padronização da linguagem C teve início em 1983 e foi finalizado apenas em 1989, com a publicação do padrão ANSI C (International Standardization Organization). Atualmente o ANSI C adota o padrão ISSO/IEC (International Standardization Organization) 9899:2011, também conhecido como C11. (JANDL JUNIOR, 2019, p.14).

Cocain (2004), relata que todas as linguagens de programação existentes possuem palavras reservadas, descrevendo que palavras reservadas são palavras que foram criadas para um propósito na linguagem de programação, como um exemplo a palavra FOR listada no Quadro 3, determina que o conteúdo após a abertura desse bloco, será repetido até o fim da lógica proposta no algoritmo. “Essas são palavras reservadas (Keywords), que em geral representam os nomes dos seus tipos, diretivas, especificadores, modificadores e alguns outros elementos da sua sintaxe”. (JANDL JUNIOR,2019, p.25).

Quadro 3 - Palavras reservada da Linguagem C

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	short	union
const	float	signed	unsigned
continue	for	sizerof	void
default	goto	static	volatile
do	if	return	while

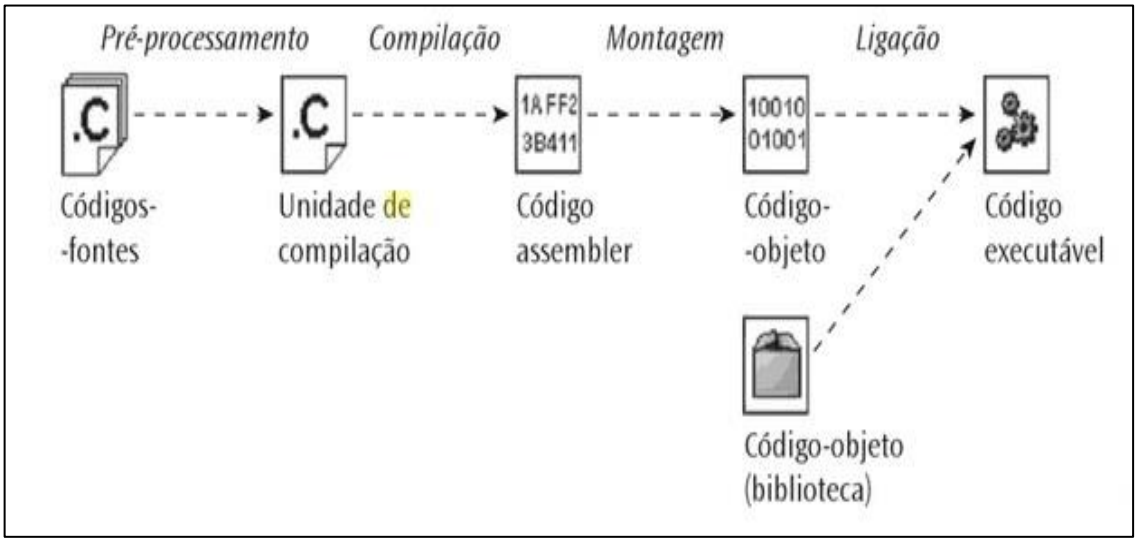
Fonte: Cocain (2004, p.33)

Conhecendo as palavras reservadas, Cocain (2004) demonstra como é a estrutura de um programa em linguagem C, além do que ele deve consistir:

- Cabeçalho do programa contendo as diretivas de compilador onde é definido o valor das constantes, variáveis e funções.
- Bloco com as instruções, denominada de função principal (main).
- Comentários no código, para consistirem na parte da documentação.

Depois de conhecido basicamente os conceitos iniciais da linguagem C, Pinheiro (2009), detalha como é o processo de compilação de programas na linguagem C, bem como demonstra em um fluxograma Figura 2 e o detalhamento no Quadro 4, de como ocorre todo o processo e seu encadeamento:

Figura 2 – Processo de compilação em C



Fonte: Pinheiro (2009, p.7)

Quadro 4 – Processo de compilação.

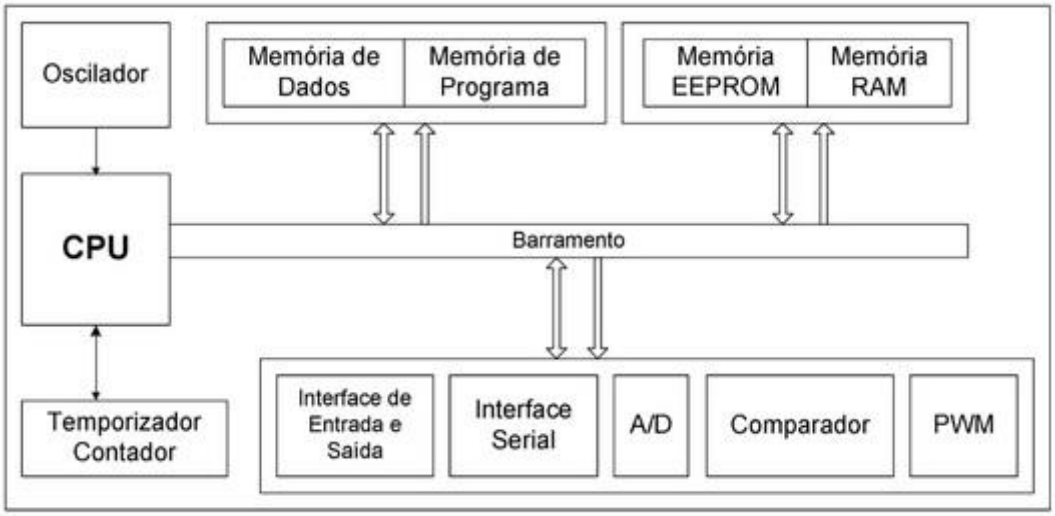
Código Fonte	Código escrito na linguagem e armazenado em um ou mais arquivos.
Código Objeto	Código gerado em linguagem de máquina para a arquitetura alvo.
Código Executável	Código gerado na linguagem de máquina com as referências resolvidas, tal qual, pode ser executado pelo processador.
Pré-Processamento	Etapa em que o código é limpo, removido espaços desnecessários e substituição de macros.
Compilação	Realizado a análise sintática e semântica da unidade, caso não possua erro se torna o código de máquina.
Montagem	Ponto em que ocorre a geração do código-objeto, em resumo código em linguagem de máquina.
Ligação	Parte final do processo, o qual junta todos os códigos objetos, criando o código executável, processo que leva o nome de compilação.

Fonte: Pinheiro (2009, p.32)

2.4 – MICROCONTROLADOR ARDUINO

Lima e Villaça (2012), descrevem que um microcontrolador é um sistema microprocessado para inúmeras funcionalidades, disponíveis em um único chip. Resumidamente um microprocessador, possui memória com programas (EEPROM), área para memória de acesso aleatório (RAM), temporizadores e Clock, todos embarcados em um único CI (Circuito integrado) demonstrado na Figura 3.

Figura 3 – Estrutura interna de um microcontrolador



Fonte: Lima e Villaça (2012, p.10)

> Dentre as funcionalidades encontradas nos microcontroladores, pode-se citar: gerador interno independente de clock, memória SRAM, EEPROM e flash, conversores analógicos-digitais (ADCs), conversores digitais-analógicos (DACs), vários temporizadores/contatores; comparadores analógicos, saídas PWM; diferentes tipos de interface de comunicação, incluindo USB, USART, I2C, CAN, SPI,JTAG, Ethernet; relógio de tempo real; circuito para gerenciamento de energia do chip; circuitos para o controle de inicialização(reset); alguns tipos de sensores; interface para LCD; e outros periféricos de acordo com o fabricante.(LIMA; VILLAÇA,2012,p.10).

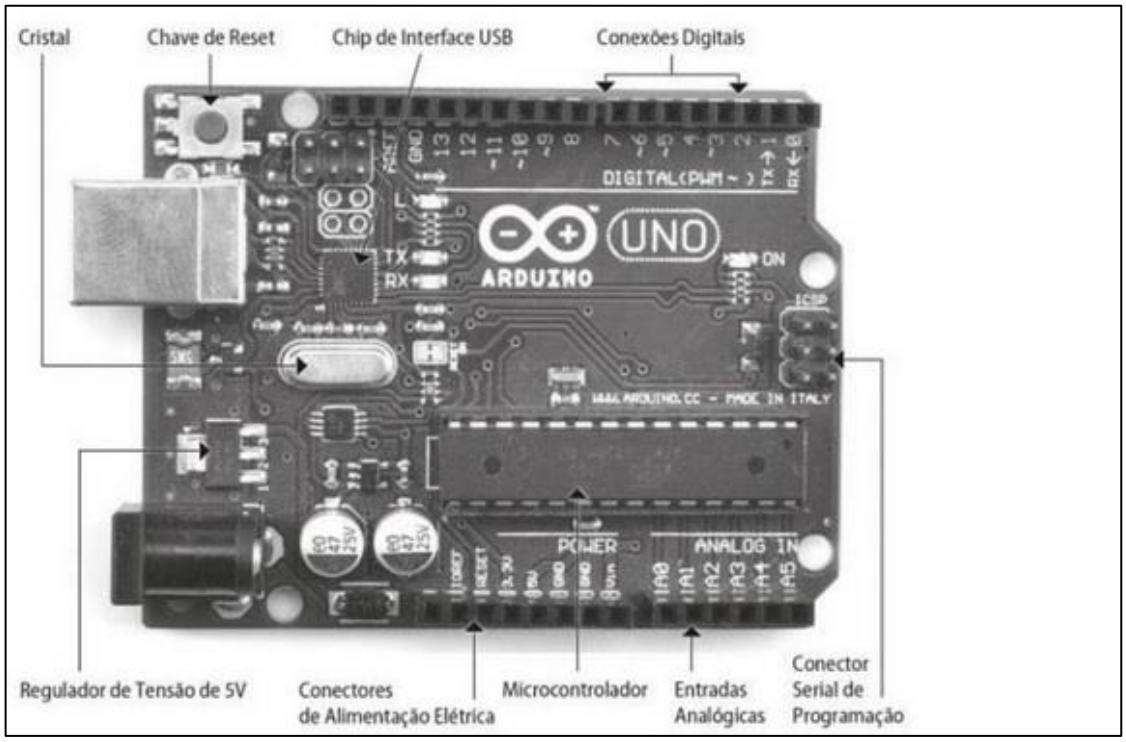
Os microcontroladores como informa Oliveira (2017), inicialmente possuíam apenas interfaces de entrada e saída(I/O), com o passar do tempo e com novas tecnologias no processo de fabricação, os chips foram ganhando novas interfaces, como por exemplo atualmente, interfaces para acesso a Ethernet, Wifi e Bluetooth.

Oliveira (2017), ainda complementa que por um microcontrolador possuir internamente todas essas funções integradas, é capaz de atender a inúmeras aplicações, dependendo de um baixo consumo de energia, podendo utilizar uma bateria como fonte de alimentação, a qual poderá manter o funcionamento do hardware por um bom tempo.

> Originalmente o Arduino foi desenvolvido para servir como recurso auxiliar no ensino de estudantes, mais adiante (em 2005), ele foi desenvolvido comercialmente por Massimo Banzi e David Cuartielles. Desde então, tornou-se um produto extremamente bem-sucedido junto a fabricantes, estudantes e artistas, devido à sua facilidade de uso e durabilidade. (MONK,2017, p.15).

Conta Monk (2017), que o Arduino se trata de uma pequena placa de projeto (*Open Source*) que contém um microcontrolador embutido em sua placa, sua forma de conexão é por uma entrada USB utilizada para comunicação com o computador, além de pinos de conexão (I/Os), que por meio destes pinos os mais variados dispositivos podem ser conectados, demonstra na Figura 4 a estrutura do Arduino.

Figura 4 – Estrutura da placa Arduino



Fonte: Monk (2017, p.12)

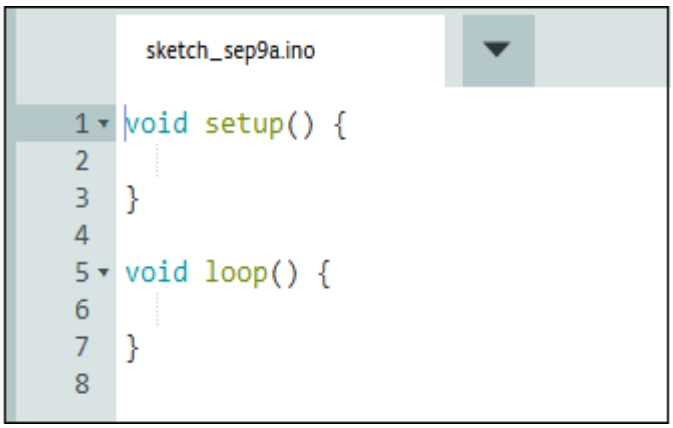
Completa ainda que o microcontrolador de uma placa Arduino é um circuito integrado (CI), que possui um total de 28 pinos, no qual está encaixado no soquete ao centro da placa, oferecendo suporte para a eletrônica digital, qual utiliza sinais lógicos em valor binário, 1 ou 0. “As placas Arduino de interface oferecem uma tecnologia de baixo custo que pode ser usada facilmente para criar projetos baseados em microcontrolador” (MONK,2017, p.11).

Pereira (2003) aborda que na criação de programas para microcontroladores e microprocessadores era uma tarefa árdua à medida que aumentava a complexidade da aplicação. Os primeiros dispositivos que podiam ser programados, necessitavam que seus programas fossem desenvolvidos em código de máquina, basicamente arquivos binários.

Monk (2017), comenta que para criar um programa em Arduino, necessitamos da ferramenta de desenvolvimento, intitulada de Arduino IDE, trata-se basicamente de um programa instalável e disponível para os principais sistemas operacionais, sendo eles Linux Windows e MacOS.

Baixada e instalada a ferramenta, basicamente para iniciar o desenvolvimento, necessitamos adicionar um novo Sketch, (Programa Vazio), onde acionamos as primeiras instruções, seguindo o padrão da linguagem de programação C, conforme a Figura 5.

Figura 5 – Estrutura base do Sketch

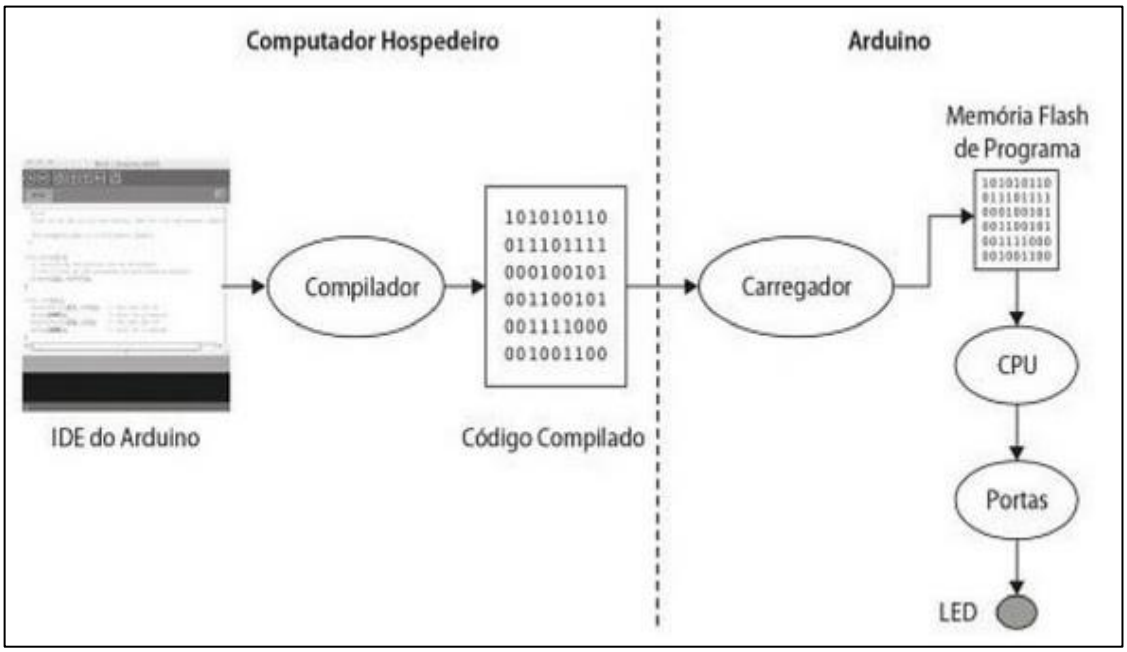


Fonte: Arduino.cc (2021)[1]

Monk (2017), complementa que, após os trechos de códigos no sketch serem feitos conforme a aplicação em desenvolvimento, o próximo passo é realizar a compilação do código para uma linguagem de máquina. Todo esse processo é realizado pela ferramenta a qual antes de transferir para o Arduino, checa se está de acordo com os padrões da linguagem, feito isso é gerado o binário e transferido para o Arduino, se encarregando de processar e mostrar o resultado, processo demonstrado na Figura 6.

[1] Disponível em: < https://store.arduino.cc/digital/create# >. Acesso em 04 dezembro, 2021

Figura 6 – Processo de compilação do Sketch



Fonte:Monk (2017, p.42)

2.5 – MICROCOMPUTADOR RASPBERRY PI

Inicialmente Oliveira (2017), aborda que o propósito do Raspberry Pi é de uma solução para computador popular, no qual, executando uma versão de sistema operacional Linux, possuindo interface gráfica, além de pacotes de aplicativos de código aberto. Por utilizar uma ampla biblioteca de softwares livres, o Raspberry Pi possibilita instalar ferramentas de desenvolvimento para as mais variadas linguagens de programação, sistemas SGBD, para gerenciamento de banco de dados, além de possuir protocolos de rede completo.

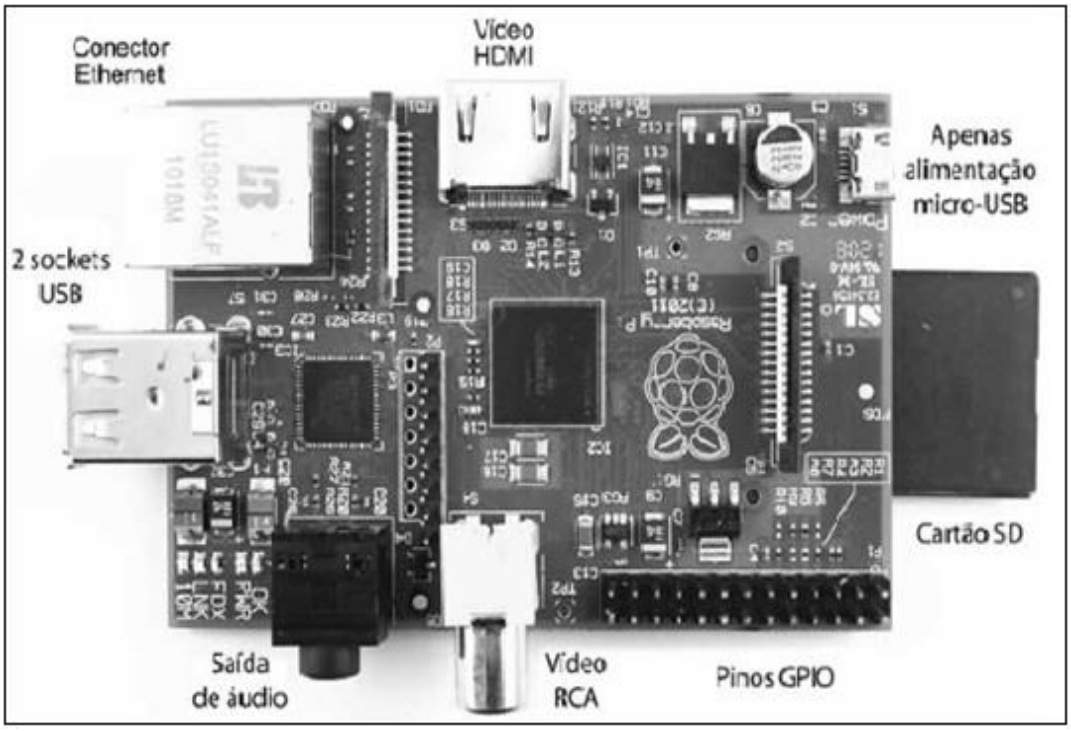
Eberman (2017 p.23), complementa que "o sistema operacional que vem integrado ao Raspberry Pi é denominado de Raspbian, uma distribuição embarcada, variante da distribuição Debian do Linux.".

> O Raspberry Pi é um pequeno computador de muito baixo custo, lançado na Inglaterra em 2012. Projetado para ser uma ferramenta de inclusão digital e pautado sobre uns grandes números de opções de programas gratuitos (softwares livres), permite que crianças e adultos tenham um primeiro contato com conceitos de computação, lógica de programação e algoritmos." (OLIVEIRA,2018, p.8).

Comentam Halfacree e Upton (2018), que a facilidade de conectar dispositivos a Raspberry Pi, é a forma que foi projetada a sua arquitetura, por seguir o padrão HAT (Hardware Attached on Top), basicamente qualquer dispositivo projetado para utilização em hardware embarcado, pode ser facilmente acoplado, usando apenas as conexões GPIOs que a placa oferece.

Oliveira (2017) detalha que, sobre o sistema Linux criado para o Raspberry Pi, o Raspbian, a placa se conecta sobre circuitos e módulos eletrônicos, denominados de GPIO (General Purpose Input/Output). Também possui portas USBs, porta Ethernet, bem como entrada para alimentação e saída para áudio e vídeo incluindo ainda seu processador central, detalhados na Figura 7, tornando uma plataforma de desenvolvimento e implementação da IoT (Internet das Coisas).

Figura 7 – Placa Raspberry Pi



Fonte: Monk, Zanolli (2013, p.19)

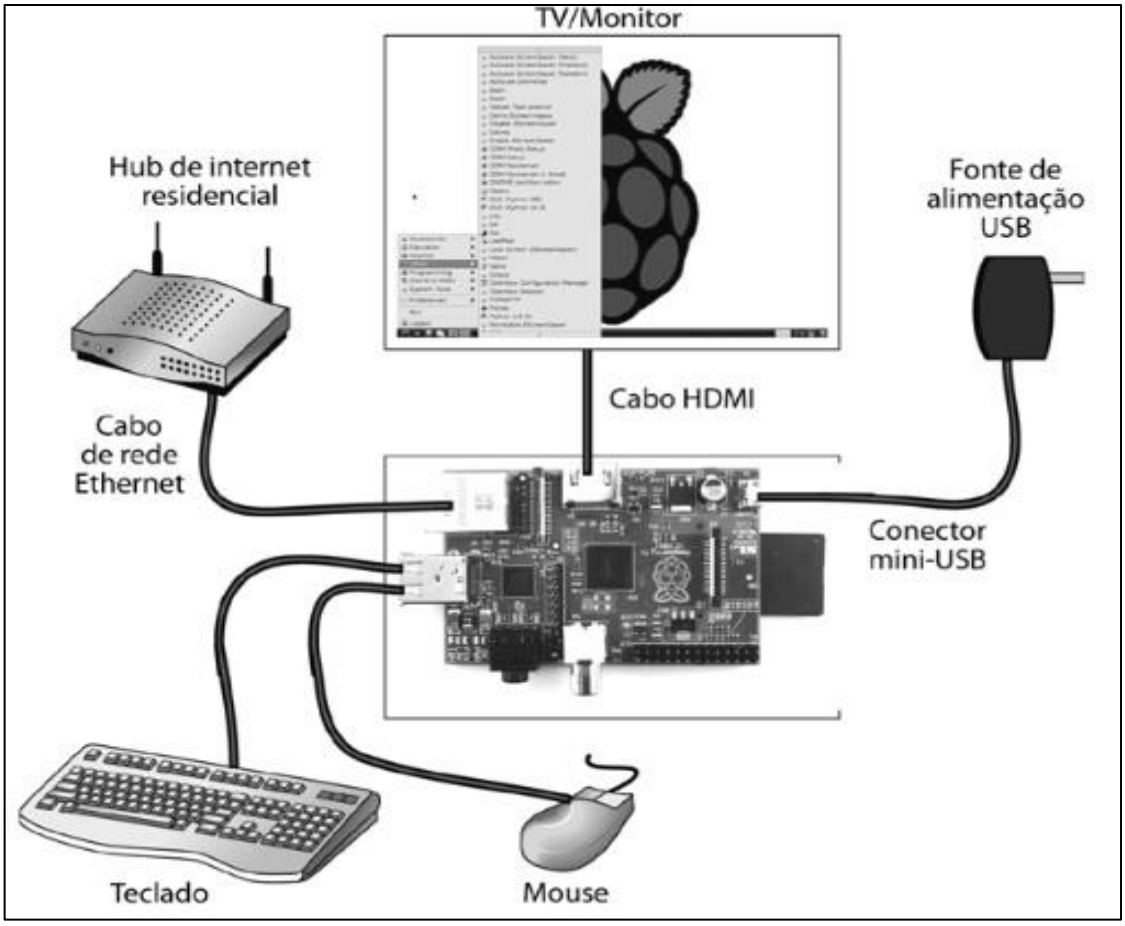
Quadro 5 – Detalhamento da Raspberry Pi

Processador /SoC	Parte central do Raspberry Pi, consiste no chip preto, conhecido como SoC (system on a Chip), processador PCM2837, arquitetura ARM Cortex A53 de 1.2 GHz.
Conector Ethernet	Possibilitar conectar o Raspberry Pi diretamente em uma rede local, sem necessidade de configuração.
Portas Usb	Composto por duas portas USB para conexão de periféricos externos, sendo eles, mouse, teclado ou outro dispositivo que utilize dessa porta.
Saída HDMI	Saída para monitores com suporte a HDMI, interface mais atual de displays, suportando resoluções até 1920x1080
Conector RCA	Saída RCA, para ser utilizada em monitores ou displays de menor resolução e mais antigos
GPIOS	GPIOS – General Purpose Input/Output
Slot SD Card	Slot para inserção do cartão de memória ao Raspberry Pi, dedicado a ler e executar o sistema nele instalado. Por padrão o Raspbian.
Porta micro USB	Porta micro USB para alimentação do Raspberry Pi
Saída de Áudio	Conector de Áudio 3,5 mm auxiliar, podendo ser conectado periféricos como caixas de som externas.

Fonte: Monk e Zanolli (2013, p.20)

Descrevem Monk e Zanolli (2013), que após os periféricos ligados, a Raspberry Pi é de fato um ambiente completo para desenvolvimento e trabalho com IoT, sendo que ainda por meio de suas GPIOs, podem sem agregados outras plataformas embarcadas como o Arduino. A Figura 8, demonstra basicamente o resultado após a montagem completa.

Figura 8 – Componentes para o ambiente de desenvolvimento com Raspberry Pi



Fonte: Monk e Zanolli (2013, p.29)

Donat (2019), complementa que comparar o Raspberry Pi a outras placas com microcontroladores, não é de fato que o Raspberry seja melhor que as demais, e sim a questão de que um Arduino ou outras placas podem ser usadas perfeitamente para projetos pequenos, assim como a Raspberry Pi, todavia, pode ocorrer uma junção de ambos, tal qual, um Raspberry Pi se encarregue de processar os dados enviado por um Arduino.

3. METODOLOGIA DA PESQUISA

O presente trabalho de conclusão de curso, é desenvolvido sobre os métodos de pesquisa aplicada, pesquisa descritiva, além da criação de protótipo. O trabalho busca apresentar uma alternativa ao produtor rural de pequeno porte. Apresentando um protótipo de automação com controle e monitoramento para auxiliar a realização de cultivo protegido em pequenas áreas.

Este trabalho tem como base o levantamento bibliográfico para montar uma base de conhecimento, tópicos na área de Linguagem de programação PHP, Banco de dados Mysql, Linguagem de programação C, além de uma abordagem sobre o microcontrolador Arduino e o Microprocessador Raspberry Pi, atualmente equipamentos com foco na utilização em IoT[2].

O foco de metodologia de pesquisa foi abordar um histórico das ferramentas que serão utilizadas no projeto, bem como um entendimento inicial de como cada uma trabalha separadamente. Desta forma, explicar como a utilização em conjunto de ambas as tecnologias relatadas podem ser utilizadas na aplicação e construção de um protótipo de automação. Além disso poder controlar e monitorar questões de temperatura, umidade, bem como posteriormente iluminação e ventilação.

Outro ponto abordado na metodologia, será a aplicação de uma avaliação qualitativa com amostragem aleatória, onde um grupo de produtores da região do Alto Vale avaliou um questionário sucinto e prático, visando ter uma noção da atual situação se tratando de automação, com a pretensão de se ter um retorno da validade do desenvolvimento do protótipo.

[2] Iot – Internet of Things (Internet das Coisas)

4. PROTÓTIPO DE AUTOMAÇÃO DE BAIXO CUSTO PARA PEQUENAS ÁREA DE CULTIVO PROTEGIDO.

Inicialmente analisou-se o mercado atual na área de estufas e monitoramento. Realizando o levantamento das informações a serem lidas e armazenadas, bem como maneiras de tratar e mostrar essas informações para o produtor. Descreveu-se um fluxo de como o protótipo vai se comportar, desde leitura das informações e de como serão exibidas, partindo desse ponto, o protótipo foi desenvolvido seguindo o fluxo determinado.

4.1 ESTADO DA ARTE

Utilizando a internet como meio de pesquisa, foi realizado a busca de produtos que oferecem suporte a monitoramento e controle. Dentre os modelos visualizados, a que mais se aproximou do protótipo é modelo DARK BOX 300, fornecido pela empresa Green Power Cultivo Indoor, no Quadro 6 é listado seus componentes principais.

Quadro 6 – Características da Dark Box 300

01	Estufa Dark Box 3000cm x 300cm x 200cm
04	Lâmpadas Vapor de Sódio 1000w
02	Lâmpadas Vapor Metálico 1000w
02	Exaustor Turbo 150mm
02	Braçadeira 150mm
01	Rolo de Fita Aluminizada 50m
04	Refletores Asa Dupla
06	Metros de Duto 150mm
04	Reatores Magnéticos 1000w
04	Temporizadores analógicos

Fonte: Green Power Cultivo Indoor,2021[3]

Analisando os principais componentes, é identificado que não há informação de componentes digitais e algo relacionado a monitoramento remoto, além disso, o produto não oferece opções de proporcionar suporte a equipamentos com fins de monitoramento por meios digitais e remotos.

[3] Disponível em:<https://www.greenpowercultivo.net.br/kits-de-cultivo/kit-dark-box-300-vapor-4000w?parceiro=9089&gclid=CjwKCAiA1uKMBhAGEiwAxzvX90hJdpKtGYJDF5bo0WUL4x99Z2CdrWbikxistyMNDdffHxE_Y9_cmRoCPKUQAvD_BwE> Acesso em: 20 novembro 2021

Outro ponto verificado é de que esse modelo em questão, é oferecido a um custo de R$ 9.666,00, sendo um valor acima da média, comparando com o custo proposto no protótipo. O Quadro 7 mostra os componentes que são oferecidos, além de do valor, calculado com a atual cotação do dólar. Outro detalhe é de que, esse protótipo pode ser adaptado a uma estrutura já existente, não sendo necessário ser adquirida, item este, opcional, deixando o valor final mais atrativo e competitivo no atual mercado.

Quadro 7 – Tabela de Custos

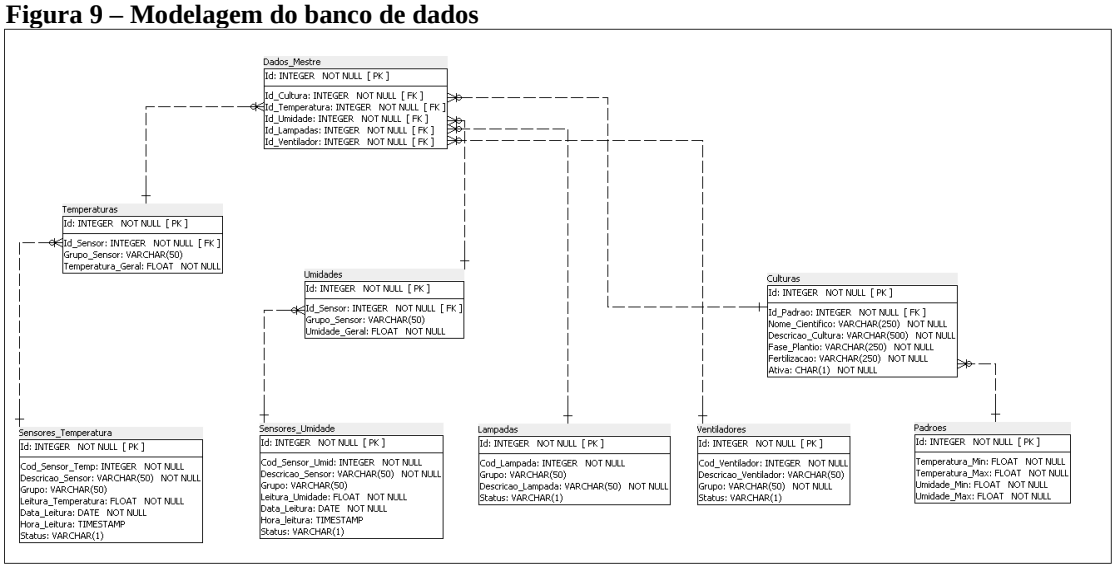
Raspberry Pi 4 – Modelo B	R$ 715,00
Arduino Uno R3	R$ 89,00
Sensor DHT11	R$ 17,00
Cabeamento MF Arduino	R$ 10,00
Protoboard	R$ 16,00
Fonte de Alimentação Arduino	R$ 41,00
Display 7’ - (Opcional)	R$ 950,00
Estrutura Metálica com cobertura – Dim 1,32M x 1,72 M – (Opcional)	R$ 650,00
Total	**R$ 2488,00**

Fonte: Acervo do Autor (2021)

4.2 MODELAGEM DO BANCO DE DADOS

Inicialmente foi proposto um modelo de banco de dados para atender a demanda das informações lidas pelo sensor. O modelo consta com nove tabelas, demonstrado na Figura 9.

Figura 9 – Modelagem do banco de dados



Fonte: Acervo do Autor (2021)

No Quadro 8, é descrito brevemente qual o papel de cada tabela no modelo proposto bem como particularidades de informações que serão armazenadas.

Quadro 8 – Detalhamento das tabelas

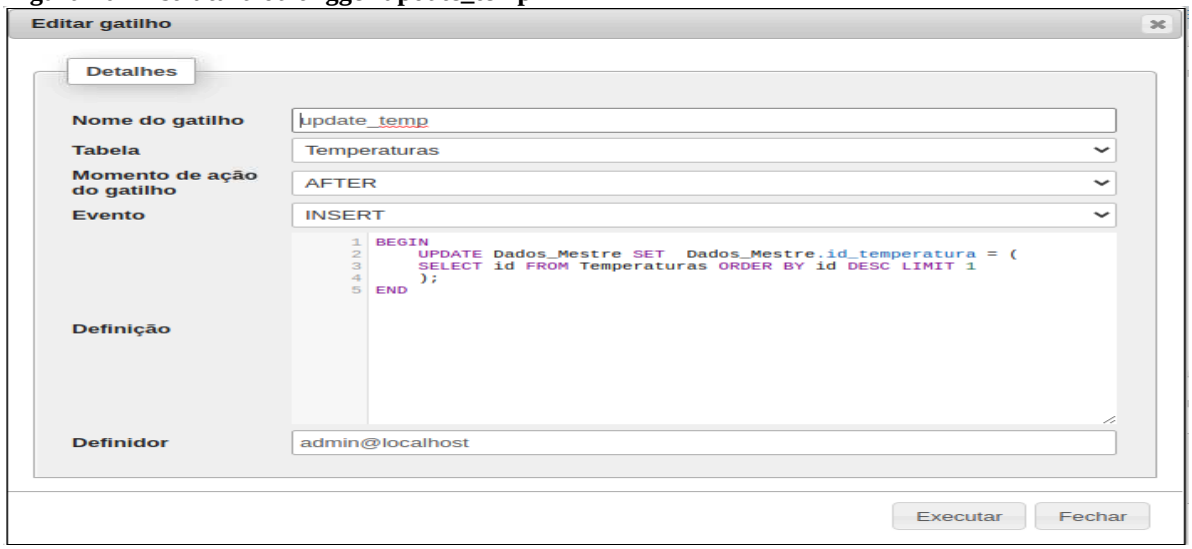
Culturas	Armazena os dados da cultura, como descrição, fase de plantio, nome científico, forma de fertilização e status de ativo
Dados_Mestre	Tabela associativa, armazena os links entre as tabelas: culturas, Temperaturas, Umidades, Lâmpadas e Ventiladores
Lampadas	Tabela para guardar as informações das lâmpadas, consiste em um código para a lâmpada, descrição, o grupo que a pertence e seu status
Padroes	Tabela responsável por guardar os padrões da cultura, campos de temperatura máxima e mínima e para umidade máxima e mínima
Sensores_Temperatura	Tabela responsável pelo cadastro dos sensores, informações de código do sensor, descrição, grupo, última temperatura lida, data da leitura, horário de leitura e status
Sensores_Umidade	Tabela responsável pelo cadastro dos sensores, informações de código do sensor, descrição, grupo, última umidade lida, data da leitura, horário
Temperaturas	Tabela responsável por gravar a temperatura lida pelo sensor do Arduino, além de gravar o id do sensor, grupo e a temperatura geral
Umidades	Tabela responsável por gravar a umidade lida pelo sensor do Arduino, além de gravar o id do sensor, grupo e a umidade geral
Ventiladores	Tabela responsável por gravar as informações dos ventiladores utilizados na estufa.

Fonte: Acervo do Autor (2021)

Descrito a parte de informações gerais do modelo da base de dados, há também a especificação de rotinas criadas, para esse modelo foi realizado a criação de duas triggers *update_temp* e *update_umid*, uma relacionada a tabela temperaturas e outra para a tabela umidade, além disso uma V*iew* chamada *dados*, criada para fins de controle e atualização das informações posteriormente.

Na Figura 10, é detalhado como a trigger na tabela temperaturas é escrita, além disso sua funcionalidade. A trigger é acionada sempre que uma nova informação é inserida na tabela temperaturas, assim que ativada, busca sempre o último *id* na tabela inserido. Essa informação é atualizada na tabela *dados_mestre*, através do comando *update* onde o *id_temperatura* anterior é substituído com o *id_temperatura* atualmente inserido.

Figura 10 – Estrutura da trigger update_temp



Fonte: Acervo do Autor (2021)

Idêntico a trigger *update_temp* a trigger *update_umid* possui a mesma função, faz a atualização na tabela *dados_mestre* sempre com o último *Id* inserido na tabela de umidades, assim mantendo sempre atualizada com atual umidade. A estrutura da *View* é composta com os campos que vão ser utilizados na apresentação da dashboard, visto que foi analisado quais os campos de maior importância para serem exibidos.

Para a criação da *View dados* foi utilizado o comando exibido na Figura 11. Consiste em usar a cláusula *create view* nome_da_view *as* comando de seleção de dados. Foi utilizado desse recurso do banco de dados, pois qualquer alteração nas informações das tabelas relacionadas a *View*, são atualizadas instantaneamente, sem precisar usar de outro recurso de manipulação.

Não foi utilizado no comando de seleção a cláusula *Left Join*, pois o modelo do banco de dados foi projetado para sempre ter um índice primário, o que torna a consulta mais dinâmica e ágil.

Figura 11 – Comando de criação da view

```
CREATE VIEW Dados AS
                  Select
                        Dm.id_cultura,
                        Cul.descricao_cultura,
                        Temp.id_sensor_temperatura,
                        Temp.temperatura_geral,
                        Umid.id_sensor_umidade
                        Umid.umidade_geral,

                        /*Lampadas*/
                         Dm.id_lampada,
                         Lan.descricao_lampada,
                         Lan.grupo As grupo_lan,
                         If(Lan.status='L','Ligada','Desligada') As status_lan,

                         /*Ventiladores*/
                         Dm.id_ventilador,
                         Ven.descricao_ventilador,
                         Ven.grupo As grupo_ven,
                         If(Ven.status='L','Ligado','Desligado') As status_ven

                         From
                            Dados_Mestre Dm,
                            Culturas Cul,
                            Temperaturas Temp,
                            Umidades Umid,
                            Lampadas Lan,
                            Ventiladores Ven

                         Where
                             ((Cul.Id_Cultura = Dm.Id_Cultura)
                           And(Temp.Id_ = Dm.Id_Temperatura)
                           And(Umid.Id = Dm.Id_Umidade)
                           And(Lan.Id  = Dm.Id_Lampada)
                           And(Ven.Id  = Dm.Id_Ventilador))

                           Order by Cul.Descricao_Cultura
                           ;
```

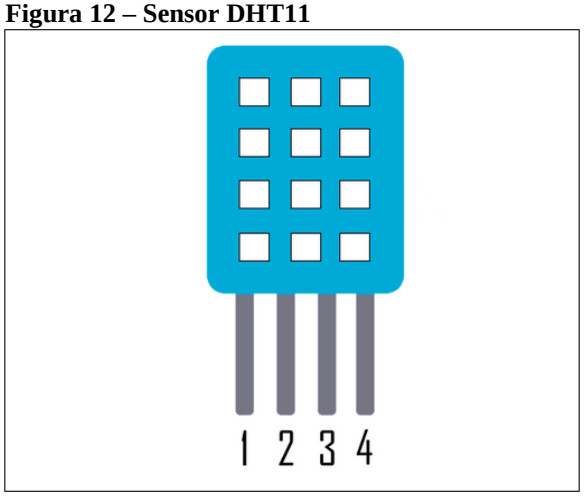
Fonte: Acervo do Autor (2021)

Neste tópico foi abordado como foi realizado a modelagem e estruturação da base de dados, atualmente visando comportar as informações necessárias para o funcionamento propostos. Visto que já foi planejado tabelas adicionais, com a ideia de possibilitar expansão em uma nova versão do protótipo.

4.3 ALGORITMO DO ARDUINO E ESTRUTURA DE MONTAGEM

Finalizada a modelagem do banco de dados, é iniciado o processo de montagem do Arduino e o sensor de temperatura e umidade. Para esse protótipo foi utilizado o sensor DHT11 Figura 12, este sensor trabalha com sinal analógico e possui uma janela de abertura para temperatura de 0° à 50°C com variação de ± 2°C e para a umidade de 20% a 90% com variação de ±5%. No Quadro 9 contém os detalhes de conexão e tensão de alimentação.

Figura 12 – Sensor DHT11



Fonte: Acervo do Autor (2021)

Quadro 9 – Informações de conexão e consumo do sensor DHT11

1 - VCC	Pino de alimentação, trabalha com tensões de 3,5V a 5V
2 - DATA	Pino para realizar a leitura de dados, trabalha com sinal analógico
3 - NC	Pino sem conexão
4 - GND	Pino para ligação do aterramento

Fonte: Acervo do Autor (2021)

O sensor DHT11 foi conectado a uma porta analógica A1 no Arduino de forma que possa ser iniciado o desenvolvimento do algoritmo responsável por iniciar a leitura da temperatura e umidade. Para iniciar o algoritmo primeiramente deve ser importar a biblioteca do sensor DHT11, como demonstrado na Figura 13.

Usando o comando *#include <DHT11.h>* (linha 1), é feito a importação da biblioteca, após a importação é necessário declarar uma definição, informando que o sensor vai se conectar a porta analógica *A1*, através do comando *#define DHTPIN A1* (linha 2).

Com a definição criada, é necessário declarar qual será o tipo do sensor, para isso é utilizado a declaração *#define DHTTYPE DHT11* (linha 4).

Para instanciar um novo sensor, é preciso passar os dados das definições, além de uma identificação (nome), utilizando a cláusula *DHT sensor01(DHTPIN, DHTTYPE)* (linha 6), desta forma o sensor está declarado e pronto para uso.

Figura 13 – Declaração do sensor

Sensores

```
1 #include <DHT.h>
2
3 #define DHTPIN A1 // Pino Analogico A1
4 #define DHTTYPE DHT11 // DHT 11
5
6 DHT sensor01(DHTPIN, DHTTYPE); // Definindo Sensor01
```

Fonte: Acervo do Autor (2021)

Com a declaração do sensor de temperatura, já é definido variáveis para controle de iluminação e de ventilação. Na Figura 14, é atribuído como tipo inteiro a variável *lamp01*, onde é definido o pino digital 13 (linha 8), para a ventilação é definido como inteiro a variável *vent01* e atribuída ao pino digital 14 (linha 9).

Para as informações de status, são criadas as variáveis *infLamp* e *InfVent* (linhas 10 e 11), ambas do tipo *char* (caractere), pois o retorno é dois status, “L” (Ligado) ou “D” (Desligado). A variável *geral* (linha 13) foi criada para unir os dados do sensor de temperatura e os status de iluminação e ventilação em uma única informação e transmiti-la.

Figura 14 – Declaração de Variáveis de controle

```
 8 int lamp01 = 13;              // Definindo Lampada01
 9 int vent01 = 14;              // Definindo Ventilador01
10 char infLamp;                 // Informação da Lampada
11 char infVent;                 // Informação do Ventilador
12
13 String geral;                 // Envio dos dados pela serialPort
```

Fonte: Acervo do Autor (2021)

Após a definição do sensor, como demonstrado na Figura 13, é iniciado o seu funcionamento, além disso, precisa ser configurado a velocidade de transmissão de dados utilizada via porta serial. Na Figura 15 é iniciado a cláusula *void setup()* (linha 15), responsável por conter a declaração do sensor, informações dos pinos que serão utilizados e a velocidade de transmissão dos dados via serial.

Inicialmente é declarado a velocidade de transmissão de dados via *SerialPort* informando o valor de 9600, padrão para comunicação serial, logo após, nosso sensor é iniciado, utilizando o comando *sensor01.begin()* (linha 17). É definido dois pinos como *outputs* (saídas digitais) (linhas 19 e 20) para uso posterior, sendo um para controle da iluminação e outro da ventilação.

Figura 15 – Setup de inicialização do sensor e porta serial

```
15 void setup()
16 {
17   Serial.begin(9600);           // Velocidade de Baudrate serial
18   sensor01.begin();             // Iniciando sensor01
19   pinMode(lamp01, OUTPUT);      // Definindo Pino13 como saida digital
20   pinMode(vent01, OUTPUT);      // Definindo Pino14 como saida digital
21 }
```

Fonte: Acervo do Autor (2021)

A cláusula *void loop()* na Figura 16 (linha 23), é responsável por executar todas as informações contidas dentro de seu bloco constantemente, nesta função são criadas duas novas variáveis do tipo *float*, uma responsável por gravar os dados de umidade e outra os dados da temperatura.

Para a leitura da umidade é definido o valor a variável *umid*, com a função *sensor01.readHumidity()* (linha 26), a temperatura recebe em sua variável *temp* os valores retornados da função *sensor01.readTemperature()* (linha 27), dessa forma as informações são recebidas e armazenadas nas duas variáveis.

Figura 16 – Cláusula void loop e variáveis de dados

```
23 void loop()
24 {
25     /*Definindo a Leitura do Sensor*/
26     float umid = sensor01.readHumidity();
27     float temp = sensor01.readTemperature();
```

Fonte: Acervo do Autor (2021)

Determinado as variáveis de temperatura e umidade, é necessário declarar uma validação aos dados atribuídos, Figura 17. É declarado a cláusula *if*, na qual por meio da função *isnan (Is Not a Number)* (linha 30), faz a validação tratando se as informações contidas são valores válidos.

Caso essa informação retorne verdadeiro, entra na condição definida e mostra a mensagem de erro, por meio do comando *Serial.println(MenErroSensor)* (linha 32), caso a validação retorne falso, seguem com o fluxo determinado.

Na sequência, outra cláusula *if* é iniciada, onde é determinado que, se a temperatura for maior que 25°C (linha 37). o programa irá informar na porta digital da lâmpada, a informação para desligar (Nível Lógico 0) e acionar a porta digital do ventilador, (nível lógico 1) (linha 39 e 40), além de encaminhar os dois status, através das *strings* “D” (desligado) e “L” (Ligado) (linha 41 e 42).

No próximo bloco da cláusula *if* (iniciado na linha 46) a sequência é contrária, caso a temperatura seja menor a lâmpada é ligada (nível lógico 1) e a ventilação é desligada (nível lógico 0) (linha 46 e 47) e os status (linha 48 e 49), permanecendo em um constante controle.

Figura 17 – Tratamento dos dados do sensor

```
29      /*Tratamento do Sensor*/
30      if (isnan(temp) || isnan(umid))
31      {
32        Serial.println(MenErroSensor);
33      }
34      else
35      {
36        //Valor padrão para ligar/desligar a ventilação/iluminacao
37        if(temp > 25)
38        {
39          digitalWrite(lamp01,LOW);
40          digitalWrite(vent01,HIGH);
41          infLamp='D';
42          infVent='L';
43        }
44        else
45        {
46          digitalWrite(lamp01,HIGH);
47          digitalWrite(vent01,LOW);
48          infLamp='L';
49          infVent='D';
50        }
```

Fonte: Acervo do Autor (2021)

Após passar na rotina de validação de dados, chega à parte de unir as informações e transmitir via porta serial. Na Figura 18 está atribuído à variável *geral* as leituras de temperatura e umidade convertidas em caractere por meio da palavra reservada *String*, concatenado com um separador traço ("-"), também é acrescentado os status da lâmpada de iluminação e o status do ventilador (linha 53). Transformado e concatenado os dados em uma única linha, é utilizado um intervalo de tempo com o comando *delay* passando o valor de 5 segundos para cada nova leitura (linha 56).

Após o intervalo de tempo, ocorre o envio da variável geral sempre em uma nova linha por meio do comando *println* (linha 57), observado que no monitor *serial* do próprio ambiente de desenvolvimento do Arduino a impressão da variável geral, é apresentada como a formatação pré-determinada anteriormente.

Figura 18 – Concatenação das variáveis

```
52      //concatenação das informações
53      geral = String(temp)+'-'+String(umid)+'-'+char(infLamp)+'-'+char(infVent);
54
55      // Realizando Impressao da Leitura
56      delay(5000);
57      Serial.println(geral);
```

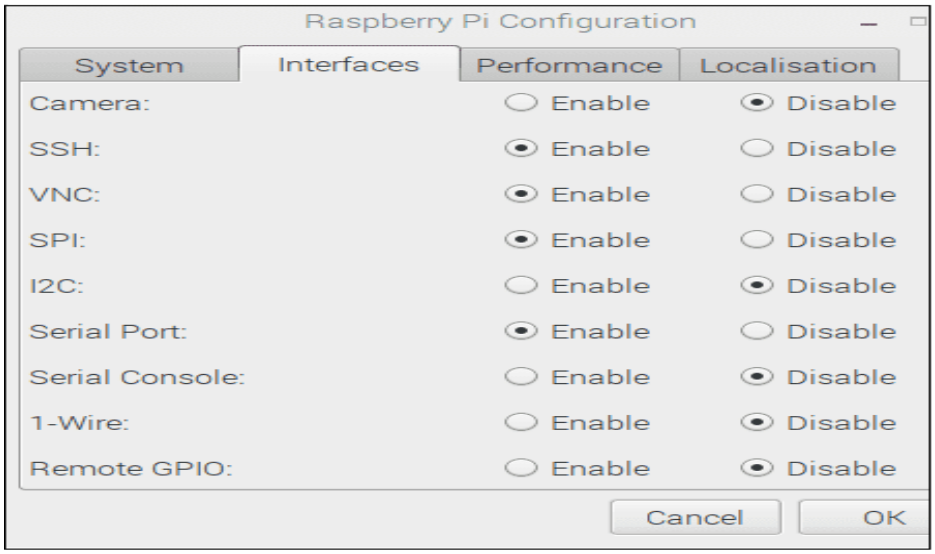
Fonte: Acervo do Autor (2021)

4.4 CONFIGURAÇÕES DO RASPBERRY PI

Para preparar o ambiente na Raspberry Pi, é necessário instalar um sistema operacional. Para o protótipo foi escolhido a distribuição Raspberry Pi OS, essa distribuição é específica para a arquitetura ARM (*Advanced RISC Machine*), trata-se de um sistema baseado na distribuição Linux Debian, além de ser robusto, oferece recursos e ferramentas por padrão.

Após a instalação do sistema no Raspberry Pi, é realizado a primeira inicialização do sistema, onde é apresentada a tela inicial e o setup para configuração e utilização. Para o protótipo, se faz necessário habilitar uma configuração para o funcionamento e comunicação usando o protocolo serial. Na Figura 19, é exibido como a configuração *Serial Port* deve ser marcada para estar ativa, as demais opções são opcionais.

Figura 19 – Configurações do Raspberry PI



Fonte: Acervo do Autor (2021)

Realizado o setup inicial de configurações, é conectado fisicamente o Arduino e a Raspberry Pi, observado na Figura 20. É opcional qual das portas USBs será conectada, ao realizar a checagem de porta USBs em uso, utilizando o comando *ls /dev/tty**, como resultado será mostrado a porta *ttyACM0* ou *ttyACM1*, confirmado esse resultado, é necessário começar a trabalhar com as bibliotecas *MySQLdb* e *Serial*.

Figura 20 – Forma de conexão Raspberry Pi e Arduino

Fonte: Acervo do Autor (2021)

Inicialmente criamos um arquivo chamado de *pySerial.py*, neste arquivo, informamos quais bibliotecas serão utilizadas pelo Raspberry Pi. Primeiramente é realizado a importação da biblioteca *serial*, posteriormente da biblioteca *Mysqldb*, ambas utilizando o comando *import*.

A biblioteca *Serial* é encarregada de realizar a leitura dos dados recebidos do Arduino, cabe a mesma regra a biblioteca *Mysqldb*, responsável por gravar as informações obtidas na porta serial e gravar no banco de dados.

Definido as importações das bibliotecas (linha 1 e 2) Figura 21, é criado a conexão com a base de dados e definições da comunicação serial. Observa-se que iniciamos uma variável *con* (linha 4), esta variável vai receber a função *Mysql.Connect*, onde é passado os dados de autenticação, posteriormente é definido o nome da base de dados (linha 5).

Com a base de dados conectada, é instanciado um *cursor* (linha 6), encarregado de realizar a leitura dos informações. Com as definições do banco de dados finalizadas, é criado a rotina para leitura dos dados via porta serial, para isso é iniciado a variável *ser*, utilizado a função *serial.Serial* para atribuir a porta USB mapeada, além disso é definido a velocidade de leitura, onde deve ser a mesma utilizada pelo Arduino, no caso é configurado com 9600 (linha 8), seguindo o padrão anterior.

Definido as questões de porta, é iniciado a leitura dos dados, para isso é definido uma variável *res* (linha 9), que recebe o valor da variável *ser* (linha 8), além da função *readline()*, que realiza a leitura de toda a informação da porta USB.

Figura 21 – Importação das bibliotecas e configurações

pySerial.py

```
1  import serial
2  import MySQLdb
3
4  con=MySQLdb.Connect('localhost','user','pass')
5  con.select_db('estufa')
6  cursor = con.cursor()
7
8  ser = serial.Serial('/dev/ttyACM0',9600)
9  res = ser.readline()
```

Fonte: Acervo do Autor (2021)

Com a realização da leitura dos dados vindos da porta USB, se faz necessário o tratamento da informação, visto que por se tratar de uma conexão serial os dados vem com caracteres a mais dos que foram enviados.

Na Figura 22, é atribuído uma nova variável denominada *dados*, onde é aplicado os valores da variável *res*, além de decodificar para o padrão *UTF8* (linha 11), dessa forma temos somente os dados necessários.

Em um teste de impressão é observado que a variável *dados* está no padrão desejado, onde a formatação fica com a estrutura (temperatura, separador, umidade, separador, status da lâmpada, separador e status do ventilador), exemplificando a saída fica (10.00-20.00-L-D).

Para a gravação no banco de dados é separado cada informação, para isso é utilizado o comando *split*, além da propriedade para pegar a posição de cada informação para guardar, para a temperatura, é criado a variável *temp*, onde recebe da variável *dados* a primeira posição antes do separador (linha 12), o mesmo processo é realizado para a umidade (linha 13), status da lâmpada (linha 14) e para o status do ventilador (linha 15), assim é gravado cada informação separada.

Antes de iniciar a gravação das informações no banco de dados, é necessário transformar os dados da temperatura e umidade para o tipo de dados float. Novamente é criado duas variáveis, uma para a temperatura, onde vai receber a variável temp do tipo *string* e será convertida para o tipo *float* (linha 18), e seguindo a mesma sequência a variável umidade (linha 19).

Após as conversões é iniciado a rotina responsável pela gravação dos dados. Para isso é definido a primeira *query* sql denominada *sqltemp*, atribuído o comando para inserir na tabela Temperaturas duas informações necessárias, qual o sensor está enviando a informação e qual a temperatura lida e os valores representados por argumentos *%s* (linha 21).

É criado mais três *querys*, uma para gravar a umidade (linha 22), outra o status da lâmpada (linha 23) e no final o status do ventilador (linha 24) ambas seguindo o mesmo padrão.

Com as querys finalizadas, é definido as informações que vão substituir os argumentos da *querys*, o primeiro argumento *argsTemp*, recebe o código do sensor e o valor da temperatura (linha 26), os demais argumentos seguem a mesma regra, com um detalhe para o último, se fez necessário utilizar uma função *rstrip(\n)* (linha 29), há casos em que o retorno do último caractere mostrava uma quebra de linha, gerando erro ao gravar a informação.

Para gravar o registro no banco de dados, é definido uma condição de repetição *while 1:*, essa condição indica que sempre vai ser repetido os comandos abaixo do bloco, neste ponto que é realizado a gravação dos dados. A primeira informação a ser gravada é a temperatura utilizado o *cursor.execute*, função encarregada de ler a *query sqltemp* e seu argumento *argsTemp (linha 32)*, assim colocando na fila de inserção, as mesmas condições são aplicadas para as *querys sqlumid* (linha 33), *sqllamp* (linha 34) e *sqlvent* (linha 35).

Com todas as *querys* e seus argumentos já gravados em memória, é disparado o comando *con.commit()* (linha 36), que trata a fila de informações previamente finalizadas e as grava no banco de dados, finalizando por completo a rotina de gravação. Todo esse processo roda em segundo plano, só terá uma parada repentina caso ocorra algum problema de comunicação entre o Arduino e a Raspberry Pi.

Figura 22 – Decodificação dos dados e gravação

```
11  dados = res.decode('utf-8')
12  temp  = dados.split("-")[0]
13  umid  = dados.split("-")[1]
14  lamp  = dados.split("-")[2]
15  vent  = dados.split("-")[3]
16
17
18  temperatura = float(temp)
19  umidade = float(umid)
20
21  sqltemp = "insert into Temperaturas (Id_Sensor_Temperatura,Temperatura_Geral) values (%s,%s)"
22  sqlumid = "insert into Umidades (Id_Sensor_Umidade,Umidade_Geral) values (%s,%s)"
23  sqllamp = "update Lampadas set status= %s"
24  sqlvent = "update Ventiladores set status= %s"
25
26  argsTemp = ("1",temperatura)
27  argsUmid = ("1",umidade)
28  argsLamp = (lamp)
29  argsVent = (vent).rstrip('\n')
30
31  while 1:
32      cursor.execute(sqltemp,argsTemp)
33      cursor.execute(sqlumid,argsUmid)
34      cursor.execute(sqllamp,argsLamp)
35      cursor.execute(sqlvent,argsVent)
36      con.commit()
```

Fonte: Acervo do Autor (2021)

4.5 – DASHBOARD DE MONITORAMENTO DOS DADOS

Finalizado a parte estrutural do protótipo, é iniciado o desenvolvimento da dashboard, para consumo das informações. Nesta parte do protótipo foi pensado em uma solução onde as principais informações serão exibidas em tela, além de possibilidade de sua manipulação, como seleção, inserção, alteração, remoção, ambos os métodos detalhados, na Figura 23.

Inicialmente realiza-se a conexão com a base de dados, utilizando as extensões *PDO* e *PDOException*, ambas são responsáveis por tratar da conexão com o banco, após informadas, inicia a classe *DataBase*, onde é definido as questões de acesso, como usuário, senha e local do banco de dados. Para isso é criado as constantes *HOST, NAME, USER, PASS* (linhas 10, 11,12 e 13), as quais são atribuídas as informações do banco de dados.

Com essa estrutura inicial finalizada, é escrita uma função para validação dos dados da conexão, denominada de *private function set connection()* linha (24), utilizando do PDO é passado uma nova instancia de conexão (linha 28), onde preenchido as constantes definidas anteriormente.

O bloco *try/catch*, realiza a validação das informações contidas na função *private function set connection()*, caso verdaceira(*true*) o banco está conectado, caso falsa(*false*) entra em uma exceção (linha 32).

Observado que nesta classe DataBase, também é checado a validação das consultas e seus argumentos, através da função *public function execute()* (linha 35). Esta função faz o mesmo controle da função de conexão, em caso de parâmetros inválidos entra na exceção (linha 44), com a estrutura alinhada o banco está conectado corretamente.

Figura 23 – Conexão com o banco de dados

Database.php

app > Db > Database.php

```php
<?php

namespace App\Db;
use \PDO;
use \PDOException;

class Database
{
    #Informações do banco de dados
    const HOST = 'localhost';
    const NAME = 'estufa';
    const USER = 'user';
    const PASS = 'pass';

    private $table;
    private $connection;

    public function __construct($table = null)
    {
        $this->table = $table;
        $this->setConnection();
    }

    private function setConnection()
    {
        try
        {
            $this->connection = new PDO('mysql:host='.self::HOST.';dbname='.self::NAME,self::USER,self::PASS);
            $this->connection->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
        }catch(PDOException $e)
        {
            die('ERRO: '.$e->getMessage());
        }
    }
    public function execute($query,$params = [])
    {
        try
        {
            $statement = $this->connection->prepare($query);
            $statement->execute($params);
            return $statement;
        }catch(PDOException $e)
        {
            die('ERRO '.$e->getMessage());
        }
    }
```

Fonte: Acervo do Autor (2021)

Nesta mesma classe de conexão é definido as rotinas de seleção, inserção, alteração e exclusão de dados, observadas na Figura 24. A função *public function insert($values)*, tem como parâmetros os campos *$filds*, *$binds* e a própria *$query*, $filds, é responsavel por conter os campos das tabelas (linha 52), $binds se encarrega de trocar os argumentos da *$query* (linha 53), que são sinalizados por “?” e por fim a *$query*, onde consta o comando de inserção no banco de dados (linha 54).

É executado a função com o comando *execute* e por fim o retorno com o último registro inserido na tabela *lastInsertId()* (linha 56). Para as demais funções, a diferença está apenas na função de seleção *public function select*, pois leva consigo três novos parâmetros, *$where, $order e $limit (linha 61,62 e 63)*, que representa a restrição de dados, ordem e limite.

Figura 24 – Rotinas SQLs para manipulação dos dados

Database.php

app > Db > Database.php

```
49
50      public function insert($values)
51      {
52          $fields = array_keys($values);
53          $binds  = array_pad([],count($fields),'?');
54          $query  ='INSERT INTO '.$this->table.' ('.implode(',',$fields).') VALUES ('.implode(',',$binds).')';
55          $this->execute($query,array_values($values));
56          return $this->connection->lastInsertId();
57      }
58
59      public function select($where = null,$order = null,$limit = null, $fields = '*')
60      {
61          $where = strlen($where) ? 'WHERE '.$where : '' ;
62          $order = strlen($order) ? 'ORDER BY '.$order : '' ;
63          $limit = strlen($limit) ? 'LIMIT '.$limit : '' ;
64          $query = 'SELECT '.$fields.' FROM '.$this->table.' '.$where.' '.$order.' '.$limit;
65
66          return $this->execute($query);
67      }
68
69      public function update($where,$values)
70      {
71          $fields = array_keys($values);
72          $query = 'UPDATE '.$this->table.' SET '.implode('=?,',$fields).'=? WHERE '.$where;
73          $this->execute($query,array_values($values));
74          return true;
75      }
76
77      public function delete($where)
78      {
79          $query = 'DELETE FROM '.$this->table.' WHERE '.$where;
80          $this->execute($query);
81          return true;
82      }
```

Fonte: Acervo do Autor (2021)

Na Figura 25, é observado como foi seguido o padrão para criação das classes que vão compor inicialmente a dashboard. Para o protótipo, é criado as classes, *Culturas, Lampadas, Mestre, SensorTemp, SensorUmid e Ventiladores*.

Para demonstrar, é instanciado a classe *SensorTemp*, além de declarar que vai utilizar a classe de conexão *DataBase*, através do comando *use*, assim como para a classe PDO. Após essa declaração, é informado a instancia *SensorTemp*, além de definir todos os campos da tabela *Sensores_Temperatura* (linhas 10 a 17). A primeira função que é criada é a de *cadastrarSensorTemp()*, essa função realiza o cadastro de um novo sensor de temperatura, retornando como válido, caso não contenha erro ao inserir.

Logo após é definido a função *atualizaSensorTemp()*, nesta função são carregados os dados já cadastrados dos sensores, possibilitando a alteração caso for necessário. É definido três outras funções, *excluirSensorTemp(), getSensorTemp() e getSensoresTemp().* Na função *excluirSensorTemp* a identificação de qual sensor a ser removido vem por passagem de parâmetro de outra rotina a ser explicada, as funções *getSensoresTemp()* e *getSensorTemp()*, são para uso edição do cadastro dos sensores e na visualização da dashboard.

Figura 25 – Padrão de criação de classe

SensorTemp.php

app > Entity > SensorTemp.php

```php
<?php

namespace App\Entity;

use \App\Db\Database;
use PDO;

class SensorTemp
{
    public $id;
    public $cod_sensor_temp;
    public $descricao_sensor;
    public $grupo;
    public $leitura_temperatura;
    public $data_leitura;
    public $hora_leitura;
    public $status;

    public function cadastrarSensorTemp()
    {
        $objDatabase = new Database('Sensores_Temperatura');
        $this-> id = $objDatabase->insert([
                                            'cod_sensor_temp'     => $this->cod_sensor_temp,
                                            'descricao_sensor'    => $this->descricao_sensor,
                                            'grupo'               => $this->grupo,
                                            'leitura_temperatura' => $this->leitura_temperatura,
                                            'data_leitura'        => $this->data_leitura,
                                            'hora_leitura'        => $this->hora_leitura,
                                            'status'              => $this->status
                                          ]);
        return true;
    }

    public function atualizarSensorTemp()
    {
        return(new Database('Sensores_Temperatura'))->update('id='.$this->id,[
                                                                'cod_sensor_temp'     =>$this->cod_sensor_temp,
                                                                'descricao_sensor'    =>$this->descricao_sensor,
                                                                'grupo'               =>$this->grupo,
                                                                'leitura_temperatura' =>$this->leitura_temperatura,
                                                                'data_leitura'        =>$this->data_leitura,
                                                                'hora_leitura'        =>$this->hora_leitura,
                                                                'status'              =>$this->status
                                                              ]);
    }

    public function excluirSensorTemp($id)
    {
        return (new Database('Sensores_Temperatura'))->delete('id='.$id);
    }

    public static function getSensorTemp($where = null,$order = null, $limit = null)
    {
        return (new Database('Sensores_Temperatura'))->select($where,$order,$limit)
                                                     ->fetchAll(PDO::FETCH_CLASS,self::class);
    }

    public static function getSensoresTemp($id)
    {
        return (new Database('Sensores_Temperatura'))->select('id='.$id)
                                                     ->fetchObject(self::class);
    }
}
```

Fonte: Acervo do Autor (2021)

A estrutura da classe *sensorTemp* já consta com todos os métodos necessários para funcionamento, porém depende do formulário de cadastro. Com a Figura 26 é mostrado como é a sua construção, inicialmente é aberto um *<form method=”POST”>* (linha 8), esse método se encarrega de enviar os dados informados a função *cadastrarSensorTemp()*, além de deixá-la preparada para a função de *editarSensorTemp()*, pois já é passado no parâmetro *value* as informações do objeto *SensorTemp* (linha 11).

Para a gravação do registro possui um botão com a propriedade *submit* (linha 47), finalizando a rotina de cadastro, para os outros formulários do protótipo, segue-se o mesmo padrão de codificação.

Figura 26 – Formulário de cadastro

formularioSensorTemp.php

includes > formularioSensorTemp.php

```
1  <main>
2      <section>
3          <a href="index.php">
4              <button class="btn btn-success"> Voltar</button>
5          </a>
6      </section>
7      <h2 class="mt-3"><?=TITLE?></h2>
8      <form method="POST">
9          <div clas="form-group mt-3">
10             <label>Codigo do Sensor</label>
11             <input type="text" class="form-control" name="cod_sensor_temp" value="<?=$objSensorTemp->cod_sensor_temp?>">
12         </div>
13         <div clas="form-group mt-3">
14             <label>Descricao do Sensor</label>
15             <input type="text" class="form-control" name="descricao_sensor" value="<?=$objSensorTemp->descricao_sensor?>">
16         </div>
17         <div clas="form-group mt-3">
18             <label>Grupo</label>
19             <input type="text" class="form-control" name="grupo" value="<?=$objSensorTemp->grupo?>">
20         </div>
21         <div clas="form-group mt-3">
22             <label>Leitura de Temperatura</label>
23             <input type="text" class="form-control" name="leitura_temperatura" value="<?=$objSensorTemp->leitura_temperatura?>">
24         </div>
25         <div clas="form-group mt-3">
26             <label>Data</label>
27             <input type="date" class="form-control" name="data_leitura" value="<?=$objSensorTemp->data_leitura?>">
28         </div>
29         <div clas="form-group mt-3">
30             <label>Hora</label>
31             <input type="datetime" class="form-control" name="hora_leitura" value="<?=$objSensorTemp->hora_leitura?>">
32         </div>
33         <div class="form-group mt-3">
34             <label>Status</label>
35             <div class="form-check form-check-inline">
36                 <label class="form-control">
37                     <input type="radio" name="status" value="L" checked>Ligado
38                 </label>
39             </div>
40             <div class="form-check form-check-inline">
41                 <label class="form-control">
42                     <input type="radio" name="status" value="D" <?=$objSensorTemp->status =='D' ? 'checked': ''?>>Desligado
43                 </label>
44             </div>
45         </div>
46         <div clas="form-group mt-3">
47             <button type="submit" class="btn btn-success">Gravar</button>
48         </div>
49     </form>
50 </main>
```

Fonte: Acervo de Autor (2021)

A Figura 27 traz em detalhes como é feito a edição das informações do sensor de temperatura, onde se faz necessário carregar a classe *SensorTemp* com o comando *use* (linha 5), após isso é realizado uma validação para verificar se o registro é diferente do ID do registro (linha 7), caso for inválido, retorna para *index.php* (linha 9), a página inicial da dashboard. Depois disso, é preciso pegar o *Id* do objeto *SensorTemp*, para isso é utilizado o método *getSensoresTemp($_GET['id'])* (linha 13), além disso é verificado se o *Id* é um *Id* válido e está registrado no banco de dados.

Figura 27 – Rotina de edição de dados

editarSensorTemp.php

```php
<?php

require __DIR__.'/vendor/autoload.php';
define('TITLE','Editar Sensor');
use \App\Entity\SensorTemp;

if(!isset($_GET['id']) or !is_numeric($_GET['id']))
{
    header('location: index.php?status=error');
    exit;
}

$objSensorTemp = SensorTemp::getSensoresTemp($_GET['id']);

if(!$objSensorTemp instanceof SensorTemp)
{
    header('location: index.php?status=erro');
    exit;
}

if(isset($_POST['cod_sensor_temp']))
{
    $objSensorTemp->cod_sensor_temp         = $_POST['cod_sensor_temp'];
    $objSensorTemp->descricao_sensor        = $_POST['descricao_sensor'];
    $objSensorTemp->grupo                   = $_POST['grupo'];
    $objSensorTemp->leitura_temperatura     = $_POST['leitura_temperatura'];
    $objSensorTemp->data_leitura            = $_POST['data_leitura'];
    $objSensorTemp->hora_leitura            = $_POST['hora_leitura'];
    $objSensorTemp->status                  = $_POST['status'];
    $objSensorTemp->atualizarSensorTemp();

    #Retorna status success e finaliza caso gravar corretamente
    header('location: index.php?status=success');
    exit;
}

#Includes das paginas necessárias
include __DIR__.'/includes/header.php';
include __DIR__.'/includes/formularioSensorTemp.php';
include __DIR__.'/includes/footer.php';
```

Fonte: Acervo do Autor (2021)

Finalizado esse processo de validação dos dados, É feito a confirmação do método post, onde valida se o campo *cod_sensor_temp* (linha 21) está preenchido, caso esteja correto, carrega as informações que sofreram alteração, executa a função *atualizaSensorTemp()* (linha 30) e retorna a página principal (linha 33).

Para a rotina de edição carregar corretamente as informações da tela, faz-se necessário carregar o cabeçalho do site *header.php* (linha 38), o formulário de cadastro *formularioSensorTemp.php* (linha 39) e por fim o rodapé da página *footer.php*, exibido na Figura 28.

Figura 28 – Rotina de exclusão

excluirSensorTemp.php

```php
<?php

require __DIR__.'/vendor/autoload.php';
use \App\Entity\SensorTemp;

if(!isset($_GET['id']) or !is_numeric($_GET['id']))
{
    header('location: index.php?status=error');
    exit;
}

$objSensorTemp = SensorTemp::getSensoresTemp($_GET['id']);

if(!$objSensorTemp instanceof SensorTemp)
{
    header('location: index.php?status=erro');
    exit;
}

$id_retorno = $_GET['id'];

if(isset($_POST['excluir']))
{
    $objSensorTemp->excluirSensorTemp($id_retorno);
    header('location: index.php?status=success');
    exit;
}
include __DIR__.'/includes/header.php';
include __DIR__.'/includes/confirmaExclusaoSensorTemp.php';
include __DIR__.'/includes/footer.php';
```

Fonte: Acervo do Autor (2021)

A exclusão, possui validações necessárias para considerar que o registro eliminado é o correto selecionado, com um detalhe, há uma variável *$id_retorno* (linha 20), na qual, salva qual o *Id* a ser removido para a função *excluirSensorTemp* (linha 24). As demais classes do protótipo seguem os mesmos padrões de codificação.

Para a criação da dashboard, foi dividido em partes. A Figura 29 demonstra como é estruturado o cabeçalho da página, onde possui apenas os dados necessários para mostrar o título, inicia-se a abertura da *div container* (linha 16), além disso é carregado o estilo de cores e formatação. Foi utilizado Bootstrap como framework para a formatação dos estilos e demais configurações por meios de arquivo CSS.

Figura 29 – Estrutura do cabeçalho da dashboad inicial

header.php

includes > header.php

```
1  <!doctype html>
2  <html lang="en">
3    <head>
4      <!-- Required meta tags -->
5      <meta charset="utf-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7
8      <!-- Bootstrap CSS -->
9      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css"
10           rel="stylesheet" integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU"
11           crossorigin="anonymous">
12
13     <title> Estufa Monitotamento</title>
14   </head>
15   <body class="bg-dark text-light">
16       <div class="container"> <!-- Abertura Div Container-->
17         <div class="jumbotron bg-primary">
18           <!--CSS para centralizar apenas os titulos-->
19           <style>
20             h1
21             {
22               text-align: center;
23             }
24             h2
25             {
26               text-align: center;
27             }
28           </style>
29             <h1>ESTUFA MONITORAMENTO</h1>
30         </div>
```

Fonte: Acervo do Autor (2021)

Para a estrutura do rodapé Figura 30, é feito o fechamento da tag *div container* (linha 1), além de também carregar o Bootstrap ,dessa forma foi terminado a estrutura da dashboard.

Figura 30 – Estrutura do rodapé da dashboard

footer.php

includes > footer.php

```
1      </div> <!-- Fechamento Div Container -->
2      <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js"
3              integrity="sha384-/bQdsTh/da6pkI1MST/rWKFNjaCP5gBSY4sEBT38Q/9RBh9AH40zEOg7Hlq2THRZ"
4              crossorigin="anonymous"></script>
5    </body>
6  </html>
```

Fonte: Acervo do Autor (2021)

O carregamento da página, Figura 31, fica a encargo do arquivo *index.php*, neste arquivo é inserido as classes responsáveis por mostrar as informações na dashboard. Usando o comando *use*, são inseridas as classes que foram estipulas no protótipo (linhas 8 a 13), posteriormente precisa carregar em variáveis distintas os dados das tabelas, neste caso foi usado as funções criadas nas classes e atribuídas as variáveis com os nomes de $listagem (linhasz 17,20,23,26,29,32) e por fim incluir o cabeçalho, a listagem e rodapé da dashboard (linhas 34,35 e 36).

Figura 31 – Estrutura do arquivo index

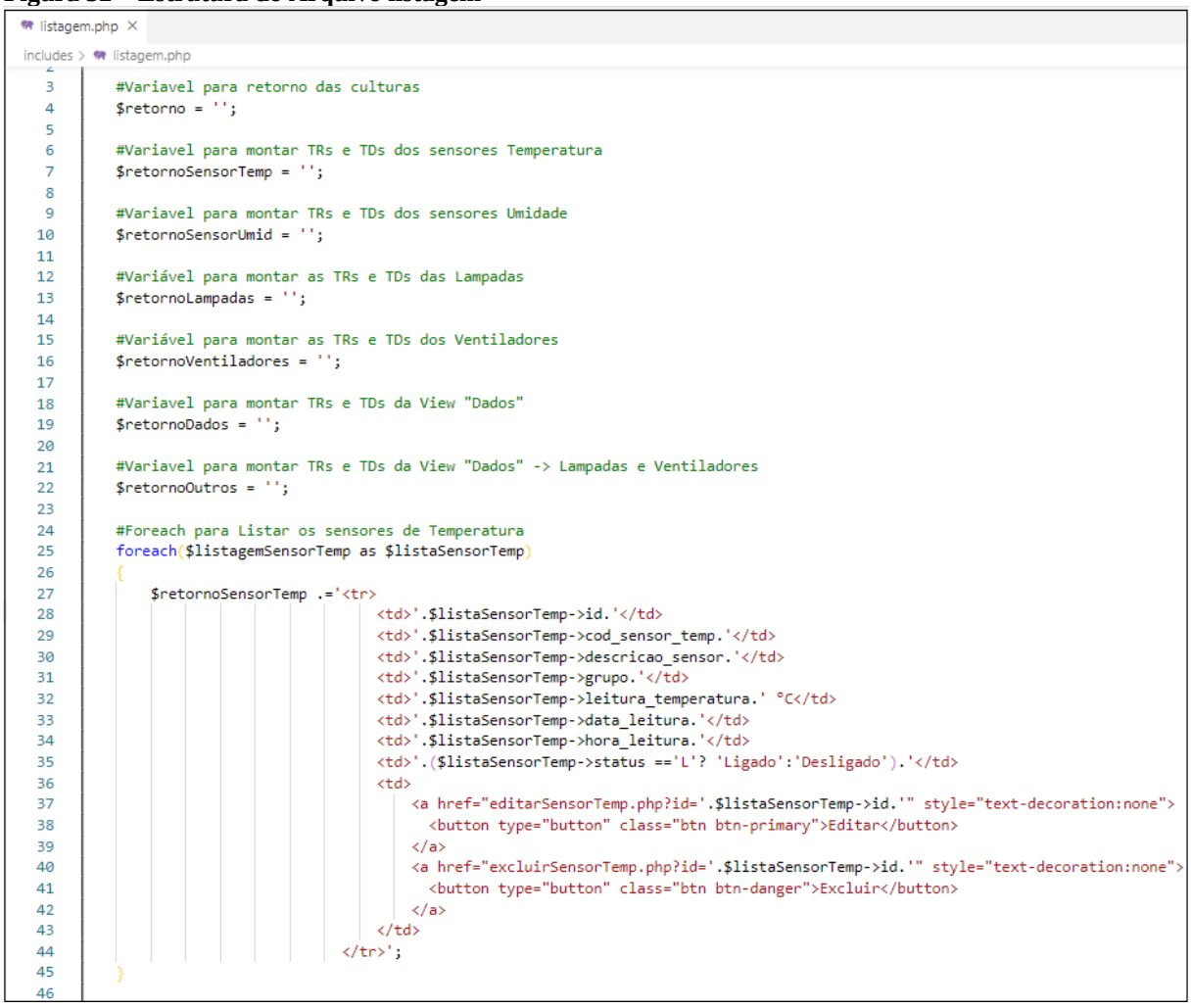
index.php

```php
<?php

require __DIR__.'/vendor/autoload.php';

/**
 * Uso das classes para montrar os resultados
 */
use \App\Entity\Mestre;
use \App\Entity\Cultura;
use \App\Entity\SensorTemp;
use \App\Entity\SensorUmid;
use \App\Entity\Lampadas;
use \App\Entity\Ventiladores;


#Variavel para trazer resultado da query (Selec * from Culturas)
$ListagemCulturas   = Cultura::getCulturas();

#Variavel para trazer resultado da query (Selec * from Sensores_Temperatura)
$listagemSensorTemp = SensorTemp::getSensorTemp();

#Vaviavel para trazer resultado da query (Selec * from Sensores_Umidades)
$listagemSensorUmid = SensorUmid::getSensoresUmid();

#Variavel para trazer resultado da query(Select * from Lampadas)
$listagemLampadas = Lampadas::getLampadas();

#Variavel para trazer resultado da query(Select * from Ventiladores)
$listagemVentiladores = Ventiladores::getVentiladores();

#Variavel para trazer resultado da query (Selec * from Dados)
$listagemDados = Mestre::getDadosMestre();

include __DIR__.'/includes/header.php';
include __DIR__.'/includes/listagem.php';
include __DIR__.'/includes/footer.php';
```

Fonte: Acervo do Autor (2021)

Com as informações já carregadas no arquivo *index.php*, na Figura 32, é realizado o carregamento dos dados exibidos na dashboard. Para isso, primeiramente é definido variáveis que são responsáveis por armazenar os dados, definidas como *$retorno* (linhas 4,7,10,13,16,19,22). Para montar os resultados dentro de uma tabela, é utilizado o comando *foreach* (linha 25), além das propriedades *HTML TR* e *TD*, para ler e carregar todas as informações.

Figura 32 – Estrutura do Arquivo listagem



```
listagem.php
includes > listagem.php
3   #Variavel para retorno das culturas
4   $retorno = '';
5
6   #Variavel para montar TRs e TDs dos sensores Temperatura
7   $retornoSensorTemp = '';
8
9   #Variavel para montar TRs e TDs dos sensores Umidade
10  $retornoSensorUmid = '';
11
12  #Variável para montar as TRs e TDs das Lampadas
13  $retornoLampadas = '';
14
15  #Variável para montar as TRs e TDs dos Ventiladores
16  $retornoVentiladores = '';
17
18  #Variavel para montar TRs e TDs da View "Dados"
19  $retornoDados = '';
20
21  #Variavel para montar TRs e TDs da View "Dados" -> Lampadas e Ventiladores
22  $retornoOutros = '';
23
24  #Foreach para Listar os sensores de Temperatura
25  foreach($listagemSensorTemp as $listaSensorTemp)
26  {
27      $retornoSensorTemp .='<tr>
28                              <td>'.$listaSensorTemp->id.'</td>
29                              <td>'.$listaSensorTemp->cod_sensor_temp.'</td>
30                              <td>'.$listaSensorTemp->descricao_sensor.'</td>
31                              <td>'.$listaSensorTemp->grupo.'</td>
32                              <td>'.$listaSensorTemp->leitura_temperatura.' °C</td>
33                              <td>'.$listaSensorTemp->data_leitura.'</td>
34                              <td>'.$listaSensorTemp->hora_leitura.'</td>
35                              <td>'.($listaSensorTemp->status =='L'? 'Ligado':'Desligado').'</td>
36                              <td>
37                                  <a href="editarSensorTemp.php?id='.$listaSensorTemp->id.'" style="text-decoration:none">
38                                    <button type="button" class="btn btn-primary">Editar</button>
39                                  </a>
40                                  <a href="excluirSensorTemp.php?id='.$listaSensorTemp->id.'" style="text-decoration:none">
41                                    <button type="button" class="btn btn-danger">Excluir</button>
42                                  </a>
43                              </td>
44                          </tr>';
45  }
46
```

Fonte: Acervo do Autor (2021)

Complementado com a Figura 33, fica a sessão principal da dashboard, onde é adicionado os botões para os cadastros, além da organização de como os dados são listados, para isso é utilizado a estrutura de tabelas, as quais são preenchidas com os dados das variáveis *$retono* (linha 186).

Figura 33 – Estrutura da visualização dos dados

listagem.php

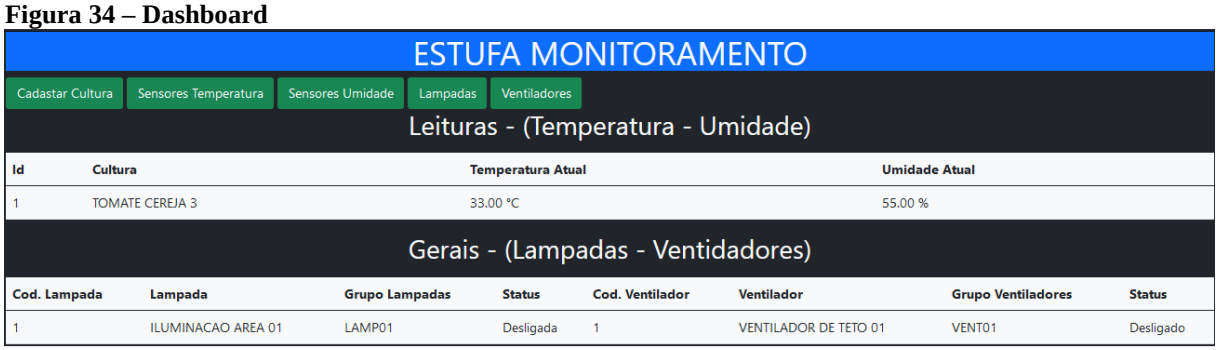
includes > listagem.php

```
153
154   <main>
155       <section>
156           <a href="cadastrarCultura.php" style="text-decoration:none" >
157               <button class="btn btn-success">Cadastar Cultura</button>
158           </a>
159           <a href="cadastrarSensorTemp.php"style="text-decoration:none">
160               <button class="btn btn-success">Sensores Temperatura</button>
161           </a>
162           <a href="cadastrarSensorUmid.php" style="text-decoration:none">
163               <button class="btn btn-success">Sensores Umidade</button>
164           </a>
165           <a href="cadastrarLampadas.php" style="text-decoration:none">
166               <button class="btn btn-success">Lampadas</button>
167           </a>
168           <a href="cadastrarVentiladores.php" style="text-decoration:none">
169               <button class="btn btn-success">Ventiladores</button>
170           </a>
171       </section>
172       <section>
173
174           <!--Dados de Temperatura e Umidade-->
175           <h2>Leituras - (Temperatura - Umidade)</h2>
176           <table class="table bg-light mt-3">
177               <thead>
178                   <tr>
179                       <th>Id</th>
180                       <th>Cultura</th>
181                       <th>Temperatura Atual</th>
182                       <th>Umidade Atual</th>
183                   </tr>
184               </thead>
185               <tbody>
186                   <?=$retornoDados?>
187               </tbody>
188           </table>
```

Fonte: Acervo do Autor (2021)

O resultado final obtido é observado na Figura 34, onde é apresentado a dashboard, listando os botões para cadastros, bem como a informação inicial, que lista em tempo real a temperatura e umidade, bem como a cultura que está sendo monitorada. Logo abaixo é visualizado as informações da lâmpada e ventilador, e seus respectivos status de operação.

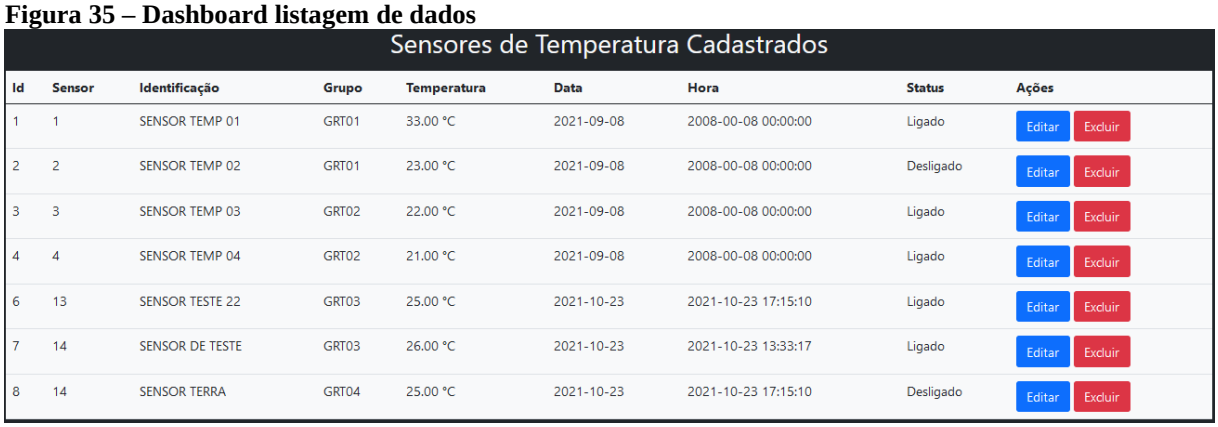
Figura 34 – Dashboard



Fonte: Acervo do Autor (2021)

Após os dados principais exibidos no topo da dashboard, logo abaixo fica a parte responsável por detalhar as informações das classes, além de fornecer as opções de edição e remoção do banco de dados, conforme a Figura 35 mostra.

Figura 35 – Dashboard listagem de dados



Fonte: Acervo do Autor (2021)

4.6 RESULTADOS

Neste tópico será apresentado o resultado da pesquisa quantitativa para aquisição de um protótipo de automação, a pesquisa foi aplicada na região do Alto Vale, tendo como seu público-alvo pequenos agricultores. Para a obtenção dos dados que serão apresentados, foi disponibilizado um questionário contendo 10 questões objetivas apresentadas no Quadro 10, com o propósito de conhecer a atual condição do pesquisado.

Esse questionário foi aplicado por meio eletrônico, utilizando a plataforma *Google Forms*, onde foi destinado a um grupo de 20 pessoas da região do Alto Vale, permanecendo disponível para resposta por 15 dias, desde seu envio. Desde a publicação e o término do prazo para respostas, obteve-se um total de 13 pessoas que efetivamente responderam, totalizando um resultado 53,85% do público-alvo alcançado, gerando resultados positivos quando a pesquisa aplicada. Os resultados por meio de gráficos que vão ser mostrados, são das questões que realmente são de suma importância para o protótipo.

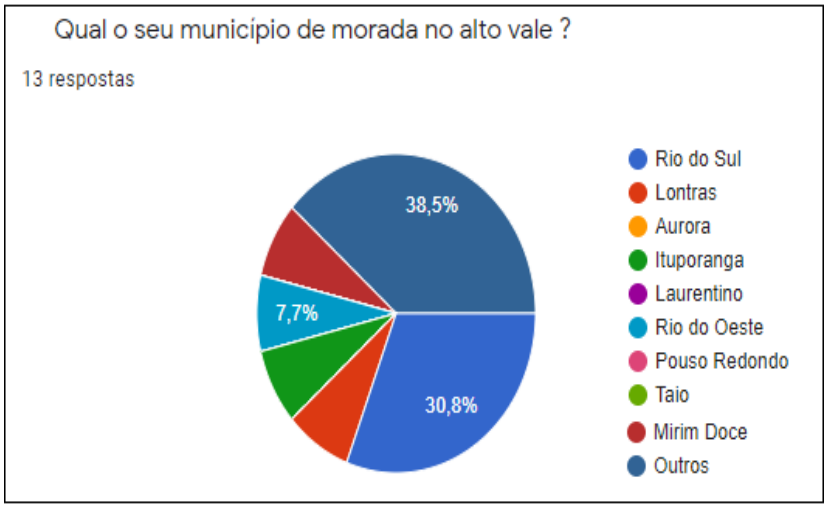
Quadro 10 – Lista de perguntas

01	Qual o seu município de morada no alto vale?
02	Você reside em área rural?
03	Qual cultivo é produzido - Exemplos (Tomate, Cebola, Alface, Fumo)?
04	Possui algum equipamento para controle e monitoramento?
05	Um produto que lhe auxilie no monitoramento e controle lhe ajudaria no dia a dia?
06	Um produto que lhe mostre os dados atuais como (temperatura, umidade, iluminação, ventilação), atenderia a sua necessidade?
07	Faria um investimento para adquirir um protótipo para testes?
08	Qual a sua opinião sobre o atual mercado de produtos que forneçam esse tipo de controle?
09	Investiria um valor estipulado em R$ 2.500 reais em um protótipo pronto para uso?
10	Indicaria esse protótipo a outro produtor caso você o adquira e tenha uma boa satisfação de uso?

Fonte: Acervo do Autor (2021)

Com o gráfico da Figura 36, obtemos uma visão geral de qual a localidade que os pesquisados estão residindo. Observa-se no gráfico que 38,5% residem no município de Rio do Sul, sendo seguido por 30,8% de outros municípios não catalogados, junto a isso segue com 7,7% para cada um dos municípios restantes.

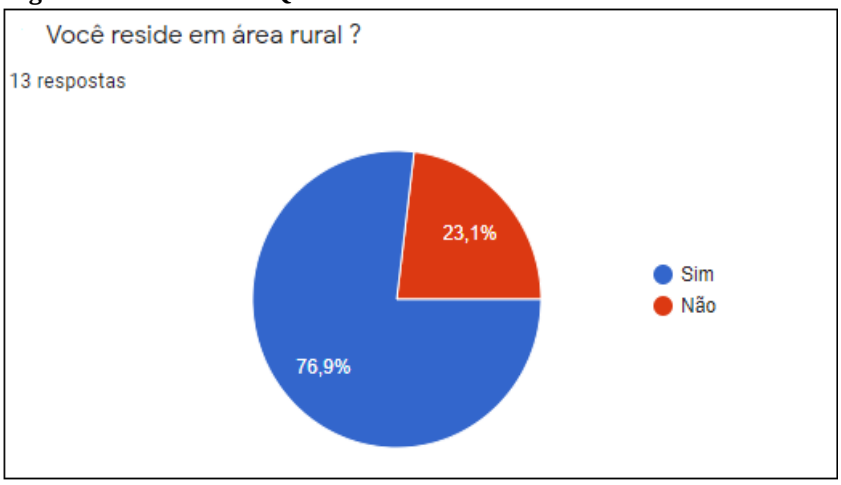
Figura 36 - Gráfico da Questão 01



Fonte: Acervo do Autor (2021)

No gráfico apresentado na Figura 37, é observado que 76,9% dos que responderem o questionário, residem em área rural, mostrando que são mais da metade que atualmente estão morando no campo, outro ponto importante para a pesquisa e o projeto.

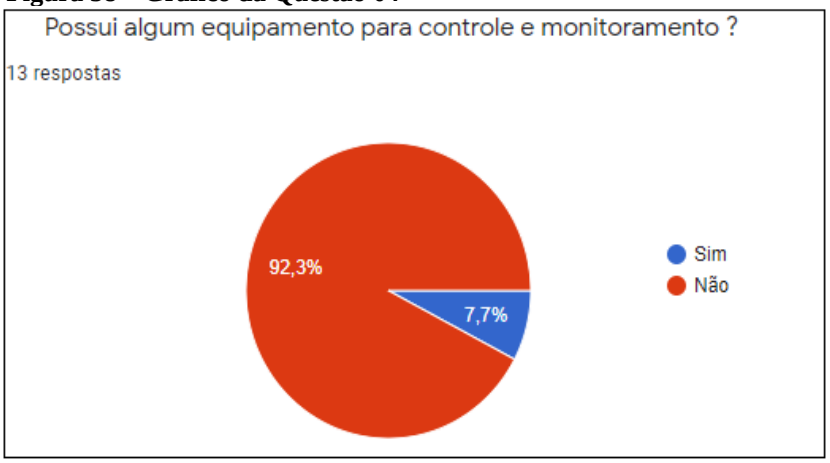
Figura 37 – Gráfico da Questão 02



Fonte: Acervo do Autor (2021)

Na apresentação do gráfico Figura 38, é realizado o questionamento se os pesquisados possuem de algum equipamento para monitoramento e controle, o foco é saber como está a difusão de tecnologias neste segmento. Nota-se que 92,3% não possuem qualquer tipo de equipamento, agregando mais um ponto positivo para o protótipo.

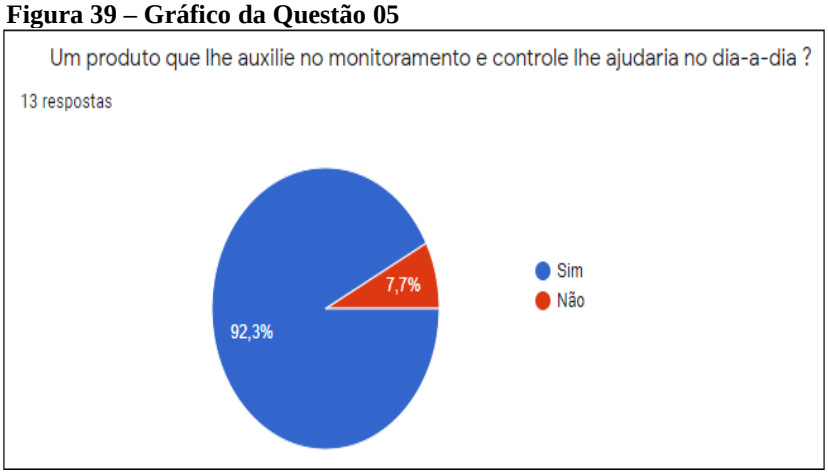
Figura 38 – Gráfico da Questão 04



Fonte: Acervo do Autor (2021)

O gráfico exibido na Figura 39, apresenta um ponto muito positivo ao desenvolvimento do protótipo, visto que 92,3 % dos pesquisados, responderam que um equipamento com essas características, vão atender perfeitamente as necessidades, demonstrando que há interesse por parte dos envolvidos em melhorar a sua qualidade de trabalho.

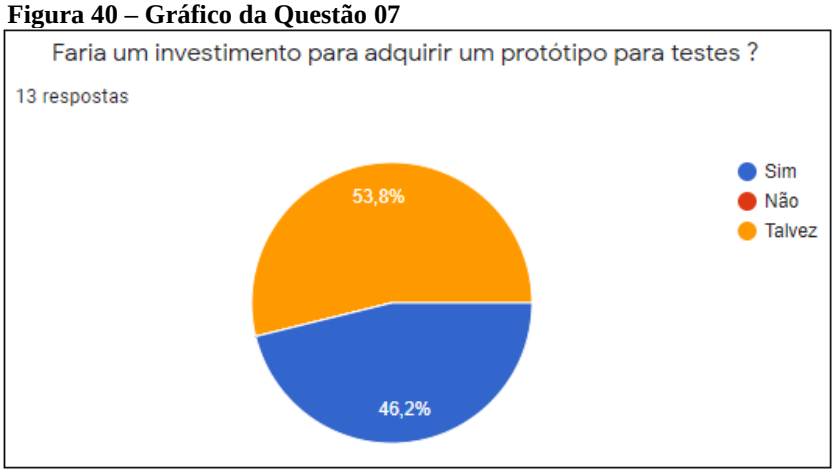
Figura 39 – Gráfico da Questão 05



Fonte: Acervo do Autor (2021)

Além disso 100% responderam à questão 05, que um protótipo que os auxilie na coleta dos dados, vai contribuir positivamente em sua rotina diária, além de proporcionar mais tempo para outras atividades. Outro fator importante analisado na Figura 40, obteve-se um total de 46,2%, que responderam que investiriam em um protótipo dessa gama, os outros 53,8% responderam que talvez fariam, claro que cabe outra análise para esse ponto, pois normalmente quem respondeu não conhece a fundo esse tipo de tecnologia, sendo necessário demonstrar em loco.

Figura 40 – Gráfico da Questão 07



Fonte: Acervo do Autor (2021)

Por fim, com os dados obtidos nesta pesquisa, foi possível identificar que há um público-alvo interessado em melhorar a sua qualidade de trabalho, outro fator observável é que, além de estarem interessados, estão dispostos a investir em um produto, o que para o desenvolvimento do protótipo agregou ainda mais valor, se comparado a outros produtos similares no mercado atual.

5. CONCLUSÃO

Com este trabalho foi descrito o processo de desenvolvimento de um protótipo de automação, com o foco para pequenas áreas de cultivo protegido, este protótipo foi desenvolvido para atender ao objetivo principal que é oferecer controle automatizado, além de proporcionar baixo custo de aquisição.

Em seu desenvolvimento foi focado atender em especifico os objetivos de criar uma rotina, para leitura de dados vindos do sensor de temperatura e umidade, utilizando o Arduino como coletor das informações, bem como o tratamento desses dados e envio, para isso foi utilizado de comunicação serial entre o Arduino e a Raspberry Pi. Todo o desenvolvimento das rotinas se deu com a utilização de bibliotecas e recursos *open source*, as quais são disponibilizadas gratuitamente, o que auxilia caso for necessário suporte ou consultas para novas implementações.

Além disso, foi proporcionado, que os dados sejam armazenados em um banco de dados instalado diretamente no Raspberry Pi, a qual tem como papel, armazenar e disponibilizar essas informações para consultas, atendendo outro dos objetivos, que é o de fornecer maneiras dessas informações serem manipuladas posteriormente. Para uma visualização amigável, foi disponibilizado um painel de acesso dashboard, onde as principais informações são exibidas, e seus dados são apresentados em tempo real, ainda oferece caso solicitado, a opção de sua customização, podendo ser alterado a ordem de como as informações são mostradas.

O protótipo ao fim de seu desenvolvimento, conseguiu cumprir com todos os objetivos propostos, além de disponibilizar recursos que não estavam no escopo do projeto, fornecendo controle para a iluminação e ventilação assim como essas informações são disponibilizadas para visualização na dashboard, tornando ainda mais afinado o seu controle, além disso, o acesso ao seu dashboard pode ser acessado por dispositivos móveis, pois o Raspberry Pi proporciona deste recurso nativamente, o que torna ainda mais cômodo a sua usabilidade e monitoramento.

5.1 TRABALHOS FUTUROS

Para trabalhos futuros, o protótipo precisa agregar em sua dashboard controle de acessos por usuário e senha, visto que no desenvolvimento, foi definido tabelas para esse controle. O protótipo pode proporcionar também que seja possível enviar informações definidas pelo usuário, como temperaturas máximas e mínimas para a cultura em cultivo, e seguindo essa mesma regra, disponibilizar o controle para a umidade. O banco de dados já foi pensado para esse propósito futuro, onde o usuário pode determinar uma janela de controle.

Além disso, possibilitar o desligamento remoto dos equipamentos, atualmente o desligamento precisa ser realizado em loco, onde em caso de problemas, necessita o usuário se deslocar ao local. Por fim, pode ser implementado um monitoramento por câmera, pois o Raspberry Pi, oferece esse recurso de forma nativa, precisando apenas a aquisição adicional de uma câmera.

REFERÊNCIAS

CARVALHO, Vinícius. **MySQL: Comece com o principal banco de dados open source do Mercado**. São Paulo: Editora Casa do Código, 2015. 170 p. ISBN 8555190800, 9788555190803

COCAIN, Luís Fernando Espindola. **Manual Da Linguagem C.** Canoas: Editora Ulbra, 2004, 500 p. ISBN 8575281070, 9788575281079

CONVERSE, Tim; PARK, Joyce. **PHP: a bíblia**. Rio de Janeiro: Campus, 2003. 868 p.

DONAT, Wolfram. **Programação do Raspberry Pi com Python: Aprenda a Programar no pequeno computador mais popular do mundo.** São Paulo: Novatec Editora Ltda, 2019, 256 p.

EBERMAM, Elivelto; et al. **Programação para leigos com Raspberry Pi.** João Pessoa: Editora IFPB, 2017, 290 p.

HALFACREE, Gareth; UPTON, Eben. **Raspberry Pi: Guia do Usuário.** Rio de Janeiro: Alta Books, 2017, 288 p.

JUNIOR, Peter Jandl. **Curso Básico da Linguagem C.** São Paulo: Editora Novatec Ltda, 2019, 232 p. ISBN 8575227327, 9788575227329

LIMA, Charles Borges De; VILLAÇA, Marco V. M. **Avr E Arduino: Técnicas de Projeto**. Florianópolis: Clube de Autores (managed), 2012, 632 p.

MILANI, André. **MySQL – Guia do Programador**. São Paulo: Novatec Editora Ltda, 2007. 400 p.

MONK, Simon. **Movimento, luz e som com Arduino e Raspberry Pi**. São Paulo: Novatec Editora Ltda, 2019, 352 p.

MONK, Simon. **Programação com Arduino: Começando com Sketches**. Porto Alegre: Editora Bookman, 2017, 200 p.

MONK, Simon; ZANOLLI, Rafael. **Programando o Raspberry PI: Primeiros Passos com Python.** São Paulo: Novatec Editora Ltda, 2013,192 p.

NIEDERAUER, Juliano. **PHP para quem conhece PHP**. São Paulo: Novatec Editora Ltda, 2008. 544 p.

OLIVEIRA, Cláudio Luís Vieira. **Raspberry Pi Descomplicado**. São Paulo: Saraiva Educação S.A, 2018, 256 p.

OLIVEIRA, Sergio de. **Internet das Coisas com ESP8266, Arduino e Raspberry Pi**. São Paulo: Novatec Editora Ltda, 2017, 240 p.

PEREIRA, Fabio. **Microcontroladores PIC: Programação em C.** São Paulo: Editora Erica Ltda, 2003, 357 p.

PETRY, Cláudia. et al. **Plantas Ornamentais aspectos para a produção 2° edição.** Passo Fundo: UPF Editora, 2008, 202 p.

PINHEIRO, Francisco de A. C. **Elementos de Programação em C.** Porto Alegre: Editora Bookman, 2009, 518 p.

RAMOS, Ricardo; SILVA, Joel da; ÀLVARO, Alexandre; AFONSO Ricardo. **PHP para Profissionais**. Rio de Janeiro: Editora Universo dos Livros, 2007. 138 p.

SAVOIA, Hugo Rossetti. **XHTML e CSS + PHP e MySQL**. São Paulo: IELD TEC Editora, 2013. 101 p.

SEBESTA, Robert W. **Conceitos de linguagens de programação.** 5. ed. Porto Alegre: Bookman, 2003. 638 p.

SOGLIO, Fábio Dal; KUBO, Rumi Regina. **Desenvolvimento, agricultura e sustentabilidade.** Porto Alegre: Editora da UFRSG, 2016, 206 p.

WEBER, Raul Fernando. **Fundamentos de arquitetura de computadores.** 3. ed. Porto Alegre: Bookman, 2008. 306 p. (Série Livros Didáticos)

APÊNDICE A

Protótipo de produto para Automação para cultivo protegido

Pesquisa de aquisição de protótipo de produto para Automação para cultivo protegido.

Resumo: A proposta deste formulário é conhecer a necessidade dos produtores da região do Alto Vale. Oferecendo um produto para auxiliar no processo de monitoramento e controle de culturas para as mais variadas culturas, além de ser intuitivo ao uso, e fornecer facilmente as informações coletada.

1 - Qual o seu municipio de morada no alto vale ?

- ○ Rio do Sul
- ○ Lontras
- ○ Aurora
- ○ Ituporanga
- ○ Laurentino
- ○ Rio do Oeste
- ○ Pouso Redondo
- ○ Taio
- ○ Mirim Doce
- ○ Outros

2 - Você reside em área rural ?

- ◯ Sim
- ◯ Não

3 - Qual cultivo é produzido - Exemplos(Tomate, Cebola, Alface, Fumo) ?

Texto de resposta longa

4 - Possui algum equipamento para controle e monitoramento ?

- ◯ Sim
- ◯ Não

5 - Um produto que lhe auxilie no monitoramento e controle lhe ajudaria no dia-a-dia ?

- ◯ Sim
- ◯ Não

6 - Um produto que lhe mostre os dados atuais como (temperatura, umidade, iluminação, ventilação), atenderia a sua necessidade ?

- ◯ Sim
- ◯ Não

7 - Faria um investimento para adquirir um protótipo para testes ?

- ◯ Sim
- ◯ Não
- ◯ Talvez

8 - Qual a sua opinião sobre o atual mercado de produtos que forneçam esse tipo de controle ?

Texto de resposta longa

9 - Investiria um valor estipulado em R$ 2.500 reais em um protótipo pronto para uso ?

Texto de resposta longa

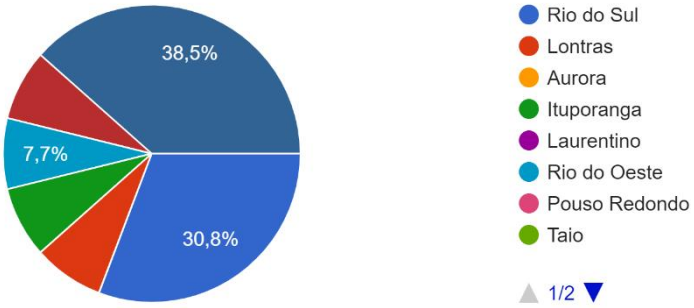
10 - Indicaria esse protótipo a outro produtor caso você o adquira e tenha uma boa satisfação de uso ?

- ◯ Sim
- ◯ Não
- ◯ Talvez

APÊNDICE B

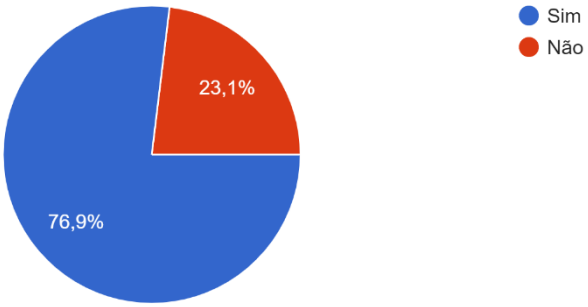
1 - Qual o seu município de morada no alto vale ?

13 respostas



2 - Você reside em área rural ?

13 respostas



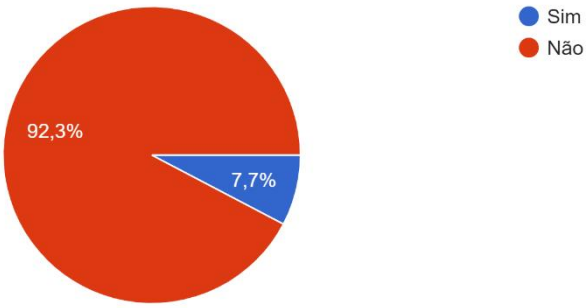
3 - Qual cultivo é produzido - Exemplos(Tomate, Cebola, Alface, Fumo) ?

12 respostas

- Milho
- Arroz
- Cebola, soja, milho, feijão
- Cebolinha e salsinha
- Cebola e fumo
- Nenhum
- Cebola fumo milho soja feijão
- Fumo
- Leite

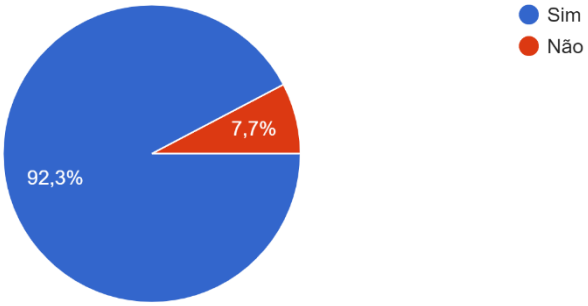
4 - Possui algum equipamento para controle e monitoramento ?

13 respostas



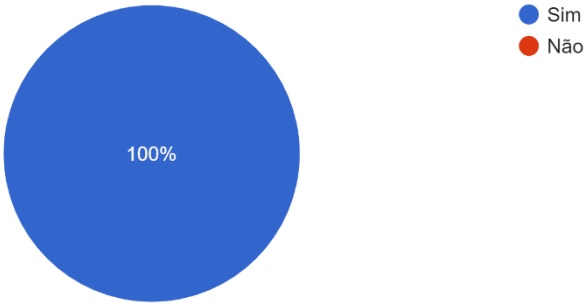
5 - Um produto que lhe auxilie no monitoramento e controle lhe ajudaria no dia-a-dia ?

13 respostas



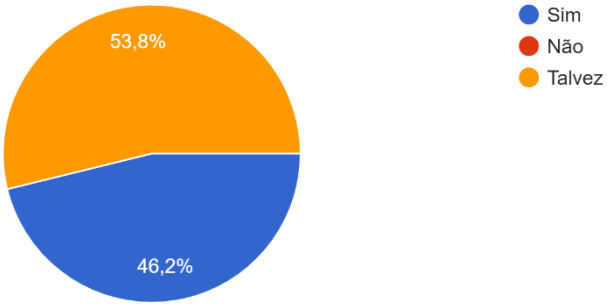
6 - Um produto que lhe mostre os dados atuais como (temperatura, umidade, iluminação, ventilação), atenderia a sua necessidade ?

13 respostas



7 - Faria um investimento para adquirir um protótipo para testes ?

13 respostas



8 - Qual a sua opinião sobre o atual mercado de produtos que forneçam esse tipo de controle ?

13 respostas

Pouco conhecimento a respeito de monitoramento.

Em geral são todos caros

Hoje o mercado não oferece produtos que de fato tenham um custo acessível com uma tecnologia de ponta.

Não se fazem presente, pois raramente ouço algo sobre.

Que eles valorizassem mais os produtos

Creio que seja interessante, com o avanço constante da tecnologia, vários produtores de pequeno e médio porte podem evoluir

Existe aparelhos que ajudam, mas com pouco manutenção, em nossa região só existe um município que "concerta" os aparelhos.

Pouco explorado

8 - Qual a sua opinião sobre o atual mercado de produtos que forneçam esse tipo de controle ?

13 respostas

Existe aparelhos que ajudam, mas com pouco manutenção, em nossa região só existe um município que "concerta" os aparelhos.

Pouco explorado

Sim, porém é muito caro

Eu cuido de gado leiteiro, tenho apenas um tanque de armazenamento onde a conferencia é feita manualmente.

Trabalho com moagem de farinha de milho em uma tafona, quando preciso ver essas informações tenho que ir até o galpão o que é ruim, dependendo dos dias

Atualmente tenho uma estufa para controle do fumo, mas ela não esta mais atualizada, preciso ir sempre no rancho para ver como esta a temperatura

Produtos e serviços muito caros

9 - Investiria um valor estipulado em R$ 2.500 reais em um protótipo pronto para uso ?

13 respostas

Sim

Dependendo de como seja a proposta e maneira de monitoramento e controle, sim investiria.

Talvez

Não sei

Se fosse necessário e os resultados forem bons, sim!

Se funcionar sim

Sim, mas negociaria o preço, talvez deixar instalado uns dias para teste.

Sim, mas negociaria o valor

Sim, é barato se atender as necessidades.

9 - Investiria um valor estipulado em R$ 2.500 reais em um protótipo pronto para uso ?

13 respostas

Dependendo de como seja a proposta e maneira de monitoramento e controle, sim investiria.

Talvez

Não sei

Se fosse necessário e os resultados forem bons, sim!

Se funcionar sim

Sim, mas negociaria o preço, talvez deixar instalado uns dias para teste.

Sim, mas negociaria o valor

Sim, é barato se atender as necessidades.

Se for valor único sem uma mensalidade obrigatória de manutenção, sim.

10 - Indicaria esse protótipo a outro produtor caso você o adquira e tenha uma boa satisfação de uso ?

13 respostas