UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

Blenda Akemi Kisine

Integração entre sistema embarcado e banco de dados no Excel aplicados ao controle automático

Sorocaba/SP

2024

Blenda Akemi Kisine

Integração entre sistema embarcado e banco de dados no Excel aplicados ao controle automático

Trabalho de Conclusão de Curso apresentado à Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba, como parte dos requisitos para obtenção do grau de Bacharela em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Ivando Severino Diniz

Sorocaba/SP

2024

K61i

Kisine, Blenda Akemi

Integração entre sistema embarcado e banco de dados no Excel aplicados ao controle automático / Blenda Akemi Kisine. -- Sorocaba, 2024

59 p. : fotos

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba

Orientador: Ivando Severino Diniz

1. Arduino (Controlador programável). 2. Excel (Programa de computador). 3. Visual Basic (Linguagem de programação de computador). 4. Diodos emissores de luz. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

Blenda Akemi Kisine

Integração entre sistema embarcado e banco de dados no Excel aplicados ao controle automático

Trabalho de Conclusão de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista (UNESP), como parte dos requisitos para obtenção do grau de Bacharel(a) em Engenharia de Controle e Automação.

Data da defesa: 22/11/2024

BANCA EXAMINADORA:

Prof. Dr. Ivando Severino DinizNome do orientador
UNESP – Instituto de Ciência e Tecnologia – Campus de Sorocaba

Profª. Dra. Maria Glória Caño de Andrade
UNESP – Instituto de Ciência e Tecnologia – Campus de Sorocaba

Prof. Dr. Felipe Paes
UNESP – Instituto de Ciência e Tecnologia – Campus de Sorocaba

Agradecimentos

Agradeço à minha família pelos carinhos, suportes incondicionais e direcionamento durante minha graduação.

Aos meus professores pelo conhecimento e orientação, principalmente aos professores Dr. Ivando Sererino Diniz, meu orientador de trabalho de conclusão de curso, Dr. Marcio Alexandre Marques, meu professor orientador de estágio, e por último e não menos importante professora Dr. Fabiane Mondini que me auxiliou em uma das disciplinas que tive DP.

Aos meus amigos por tudo suporte, companheirismo e apoio, sem eles nada disso seria possível, principalmente durante a pandemia, como a Fernanda Carvalho, Anna Dias, Johnnatan Rodrigues, Milene da Silva e Bianca Themoteo. Tivemos muitos momentos, repletos de risadas, diversão e até algumas tristezas, mas juntos conseguimos superar tudo. Cada encontro, cada conversa, tornou meus dias mais leves e significativos, que estará guardado na minha memória, tenho certeza de que esse contato vai perdurar por muitos anos.

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema de automação que integra a Microsoft Excel, por meio de programação VBA (Visual Basic for Applications), com o microcontrolador Arduino para o acionamento de LEDs na cor amarelo, verde e vermelho. O objetivo é explorar as capacidades do Excel como ferramenta de armazenamento e manipulação de dados em projetos de baixo custo, além de avaliar sua viabilidade como uma solução complementar para banco de dados em aplicações de automação, e também a integração entre o Excel e o Arduino. Através de um formulário interativo no Excel, os dados são inseridos, atualizados e enviados ao Arduino, por meio de um script em Python, que aciona diferentes LEDs conforme as especificações dos valores recebidos. A comunicação entre o Excel e o Arduino é intermediada por um script em Python, que permite a troca de dados via porta serial, proporcionando uma resposta visual imediata. Diversos testes foram realizados para garantir a estabilidade e precisão do sistema. Os resultados demonstram que a combinação entre Excel, VBA e Arduino oferece uma solução prática para projetos simples de automação e controle de hardware, contribuindo para o entendimento das limitações e potencialidades dessas ferramentas em um contexto de automação e manipulação de dados.

Palavras-chave: Microsoft Excel, VBA, Arduino, controle de LEDs e integração de sistemas.

ABSTRACT

This work presents the development of an automation system that integrates Microsoft Excel, through VBA (Visual Basic for Applications) programming, with the Arduino microcontroller for controlling yellow, green, and red LEDs. The objective is to explore Excel's capabilities as a tool for data storage and manipulation in low-cost projects, as well as to evaluate its feasibility as a complementary database solution in automation applications and the integration between Excel and Arduino. Through an interactive form in Excel, data is entered, updated, and sent to the Arduino via a Python script, which activates different LEDs based on the specifications of the values received. The communication between Excel and Arduino is facilitated by a Python script that enables data exchange via the serial port, providing an immediate visual response. Various tests were conducted to ensure the system's stability and accuracy. The results demonstrate that the combination of Excel, VBA, and Arduino offers a practical solution for simple automation and hardware control projects, contributing to an understanding of the limitations and potential of these tools in the context of automation and data manipulation.

Keywords: Microsoft Excel, VBA, Arduino, LED control, and system integration.

LISTA DE ILUSTRAÇÕES

Figura 1 – Planilha do Excel 17
Figura 2 – Interface do VBA 19
Figura 3 – Interface do VBA para a criação do formulário 19
Figura 4 – Dispositivo Arduino Uno 21
Figura 5 – Interface do Arduino 21
Figura 6 – Planilha do Excel recebendo os dados dos sensores conectados ao Arduino 23
Figura 7 – Programação no VBA do Excel 23
Figura 8 – Botão para inserir manualmente os dados 25
Figura 9 – Tela do VBA para a construção do formulário 26
Figura 10 – Interface do formulário 26
Figura 11 – Arquivo “Projeto.xlsm” 29
Figura 12 – Arquivo “Gerenciador dados.xlsm” 29
Figura 13 – Local dos arquivos no computador 30
Figura 14 – Código sketch desenvolvido no Arduino 31
Figura 15 – Circuito desenvolvido para o acionamento dos LEDs. 32
Figura 16 – (a) Acionamento do LED na cor vermelho (b) LED apagado após 10 segundos 34
Figura 17 – (a) Acionamento do LED na cor amarelo (b) LED na cor vermelho aceso (c) LED na cor vermelho apagado após 10 segundos 35
Figura 18 – (a) Acionamento do LED na cor verde (b) LED na cor vermelho aceso (c) LED na cor vermelho apagado após 10 segundos 36
Figura 19 – Inserção de valores no formulário para validação da programação 38
Figura 19 – Fluxograma do processo do projeto 39

LISTA DE ABREVIATURAS

IDE *Integrated Development Environment*

IoT *Internet of Things*

ODBC *Open Database Connectivity*

SGBDs Sistema de Gerenciamento de Banco de Dados

SQL *Structured Query Language*

SUMÁRIO

1 **INTRODUÇÃO** **11**
2 OBJETIVO **15**
3 REVISÃO BIBLIOGRÁFICA **16**
3.1 Banco de dados e SQL 16
3.2 Microsoft Excel como ferramenta complementar de banco de dados 17
3.3 A integração de Excel e Arduino para sistemas interativos 19
3.4 Análise de casos práticos no GitHub 22
4 MATERIAIS E MÉTODOS **25**
4.1 Materiais 25
4.2 Métodos 25
4.2.1 VBA 25
4.2.2 Arduino 30
5 RESULTADOS E DISCUSSÕES **37**
6 CONCLUSÃO **41**
REFERÊNCIAS BIBLIOGRÁFICAS **43**
APÊNDICE A – Código completo desenvolvido por meio do VBA do Excel (a) Formulário (b) Módulo 1 (c) Módulo 2 (d) Módulo 3 ..... **46**
APÊNDICE B – Script Python **57**
APÊNDICE C – Código sketch do Arduino **58**

1 INTRODUÇÃO

A definição de banco de dados é uma coleção de dados inter-relacionados, que não possui tamanho nem complexidade definidos, tudo depende da utilização e das necessidades específicas do usuário ou organização (Elmasri; Navathe, 2005).

Para a construção de um banco de dados, é necessário armazenar os dados em local seguro e contar com um sistema, como o sistema gerenciador de banco de dados (SGBD), que tenha controle e acesso para ser capaz de realizar processos de construção, compartilhamento e manipulação de bancos de dados entre os usuários e as aplicações realizadas (Machado, 2014).

A linguagem de programação padronizada utilizada em banco de dados é chamada de linguagem de consulta estruturada (SQL). Ela permite que usuários e desenvolvedores interajam com os dados armazenados em SGBD por meio de comando e consultas (Machado, 2014).

Um banco de dados está mais presente no cotidiano da sociedade, por exemplo em uma transação bancária, compras *on-line*, passagens aéreas e entre outras situações (Elmasri; Navathe, 2005). O artigo de Petrov *et al.* (2022) exemplifica o funcionamento de banco de dados em relação a digitalização dos serviços educacionais. Além de utilizar vários bancos de dados em um servidor, o artigo ressalta que para garantir acessibilidade e produtividade, é fundamental desenvolver uma estratégia de administração que maximize o acesso, que exista a possibilidade de rápida recuperação de falhas e minimize o tempo de inatividade, mantendo a segurança. Essa estratégia deve incluir servidor dedicado, criptografia e configurações que respeitem os padrões de segurança e gerenciamento de banco de dados (Petrov *et al*., 2022).

A demanda crescente por habilidades em banco de dados no mercado de trabalho, impulsionada pela transformação digital e pelo aumento da *big data*, reforça a importância dessa competência, posicionando-a como um diferencial competitivo tanto para indivíduos quanto para organizações (Chaudhuri *et al*., 2011; García *et al*., 2020).

O Microsoft Excel é um software de planilha eletrônica amplamente utilizado que faz parte do pacote Microsoft Office. Inicialmente, foi projetado para facilitar cálculos e análises de dados, contudo o Excel evoluiu ao longo dos anos, tornando-se uma ferramenta multifuncional para a organização, manipulação e análise de dados. A interface do Excel é amigável e as funcionalidades contidas nele são robustas, tornando-o uma opção popular tanto para usuários individuais quanto para empresas (Walkenbach, 2015).

Embora o Excel não seja um sistema de gerenciamento de banco de dados (SGBD) como o SQL Server ou MySQL, ele pode ser utilizado como um banco de dados simples em situações que não exigem a complexidade e a escalabilidade de um SGBD, como formulário e interoperabilidade. O Excel pode ser uma solução viável para pequenas organizações que precisam de um banco de dados em situações específicas, ele não substitui a funcionalidade e a segurança oferecidas por SGBDs dedicados em ambientes que exigem maior robustez e controle sobre os dados. Essa nuance é importante para entender as capacidades e restrições do Excel como uma ferramenta de gerenciamento de dados (Walkenbach, 2015).

A integração entre um banco de dados simples e Excel representa uma habilidade crucial no contexto atual de análise de dados, oferecendo vantagens significativas para usuários e empresas de pequenos e médios portes. Essa combinação permitem a extração e análise eficiente de dados, facilitando a automação de relatórios dinâmicos e a consolidação de informações de diferentes fontes quando é utilizada de maneira conjunta ao Power BI, por exemplo. Além disso, a utilização de Excel, uma ferramenta amplamente acessível, democratiza o acesso à análise de dados, capacitando profissionais não especialistas em banco de dados (Chaudhuri *et al*., 2011; García *et al*., 2020).

O avanço das tecnologias digitais permitiu que ferramentas antes limitadas ganhassem novas aplicações. Um exemplo notável é o Microsoft Excel, que com o auxílio do *Visual Basic for Applications* (VBA) agora possibilita a criação de sistemas automatizados e controle de hardware, como o Arduino. O Arduino, uma plataforma de hardware de código aberto, é amplamente adotado em projetos de automação e prototipagem eletrônica por sua simplicidade e versatilidade, sendo

utilizado em diversas áreas, como educação. O Arduino é composto por uma placa de circuito que integra um microcontrolador e portas de entrada e saída, ele possibilita o controle de uma ampla gama de dispositivos, como LEDs, motores, sensores e atuadores (Banzi; Shiloh, 2014).

O ambiente de programação do Arduino, *Integrated Development Environment* (IDE), baseado em uma variante simplificada de C++, oferece uma curva de aprendizado amigável para iniciantes, facilitando a criação de sistemas interativos. A plataforma é especialmente popular por sua capacidade de integrar-se com as outras ferramentas e softwares, como o Python, por meio da comunicação serial, permitindo que o microcontrolador responda a comandos externos em tempo real e seja utilizado em sistemas mais complexos, que requer a conexão entre hardware e software (Monk, 2013).

Python é uma linguagem de programação de alto nível amplamente adotada devido à sua flexibilidade, o que permite sua aplicação de uma variedade de contextos, desde ciência de dados e aprendizado de máquina até automação de sistemas e controle de dispositivos eletrônicos. Com uma sintaxe intuitiva e uma vasta biblioteca, a biblioteca pySerial, por exemplo, possibilita o envio e o recebimento de dados pela porta serial, viabilizando a integração de sistemas baseados em hardware com softwares. Essa capacidade de conectar-se e controlar dispositivos externos torna o Python uma linguagem ideal para aplicações que demandam interatividade e reposta a eventos físicos em tempo real (Al Sweigart, 2015). Além disso, o Python é valorizado pela comunidade de desenvolvedores e acadêmicos, o que resulta em uma grande variedade de documentação e suporte, facilitando seu uso em projetos (Vanderplas, 2016).

Dessa forma, este trabalho propõe explorar a implementação de um sistema que utiliza o Excel para receber dados a partir de um formulário, o envio de dados para o Arduino acontece por meio de um script em Python, e o Arduino por fim executa o acionamento de LED de acordo com a faixa estabelecida no projeto, destacando as possibilidades de interatividade e automação em processos de coleta e análise de dados, e também as capacidades e limitações do Excel em relação ao banco de dados. O problema de pesquisa que orienta este trabalho é: quais são as melhores práticas para integrar o Arduino e o Excel?

Para isso, será realizada uma revisão da literatura existente e uma análise de casos práticos, utilizando os projetos disponibilizados na plataforma GitHub que contenham casos práticos relacionados ao tema em estudo, a escolha de busca de casos práticos na plataforma GitHub se justifica pela escassez de literatura científica que aborde o tema de forma aplicada. Dessa forma, a pesquisa buscará identificar exemplos implementados na comunidade de desenvolvimento, proporcionando um olhar prático e complementar às discussões teóricas existentes.

2 OBJETIVO

O projeto tem como objetivo contribuir para um melhor entendimento das capacidades e limitações da ferramenta Excel em relação ao banco de dados tradicionais, avaliando se o Excel atualmente pode ser considerado uma solução viável para o armazenamento, manipulação de dados e segurança. E investigar como o Excel pode ser utilizado como uma interface simples de visualização e armazenamento de dados em sistemas automatizados.

Além disso, o projeto visa explorar a integração entre o Excel e Arduino, demonstrando como essas ferramentas podem ser combinadas para criar sistema automatizado que armazena os dados no Excel e de acordo com os dados inseridos é acionado um LED no Arduino. Para projetos que requerem uma solução de baixo custo e sem necessidade de infraestrutura de banco de dados mais avançada, visto que será integrado o Excel com o Arduino.

3 REVISÃO BIBLIOGRÁFICA

3.1 Banco de dados

A definição de banco de dados é uma coleção estruturada de dados que permite seu armazenamento e manipulação de maneira organizada. A principal função de um banco de dados é garantir a integridade das informações, mesmo com múltiplos acessos simultâneos por usuários ou sistemas diferentes. A complexidade de um banco de dados pode variar desde uma planilha básica até sistemas robustos com milhões de registros distribuídos em servidores, como um sistema hospitalar que gerencia agendamento de consultas, controle de estoque de medicamentos, históricos médicos de pacientes dentre outras informações pertinentes ao hospital, médicos, enfermeiros e pacientes (Elmasri; Navathe, 2005).

Os bancos de dados são gerenciados por Sistemas Gerenciadores de Banco de Dados (SGBDs), que oferecem uma interface para inserir, consultar, atualizar, e deletar dados. Nos SGBDs tem que garantir segurança, controle de acesso, recuperação de falhas e replicação de dados. Os sistemas gerenciadores de banco de dados mais conhecidos e utilizados são MySQL, SQL Server e PostgreSQL (Elmasri; Navathe, 2005).

A *Structured Query Language* (SQL) traduza como linguagem de consulta estruturada é a linguagem padrão utilizada para consultar e modificar dados armazenados em SGBDs. Comandos SQL permite operações básicas como inserção (INSERT), consulta (SELECT), atualização (UPDATE) e remoção (DELETE) de dados. Essa linguagem é adotada por ser versátil e capaz de lidar com grandes volumes de dados, além de garantir consistência em transações críticas, como por exemplo em sistemas de comércio eletrônico e financeiros (Chaudhuri et al., 2011).

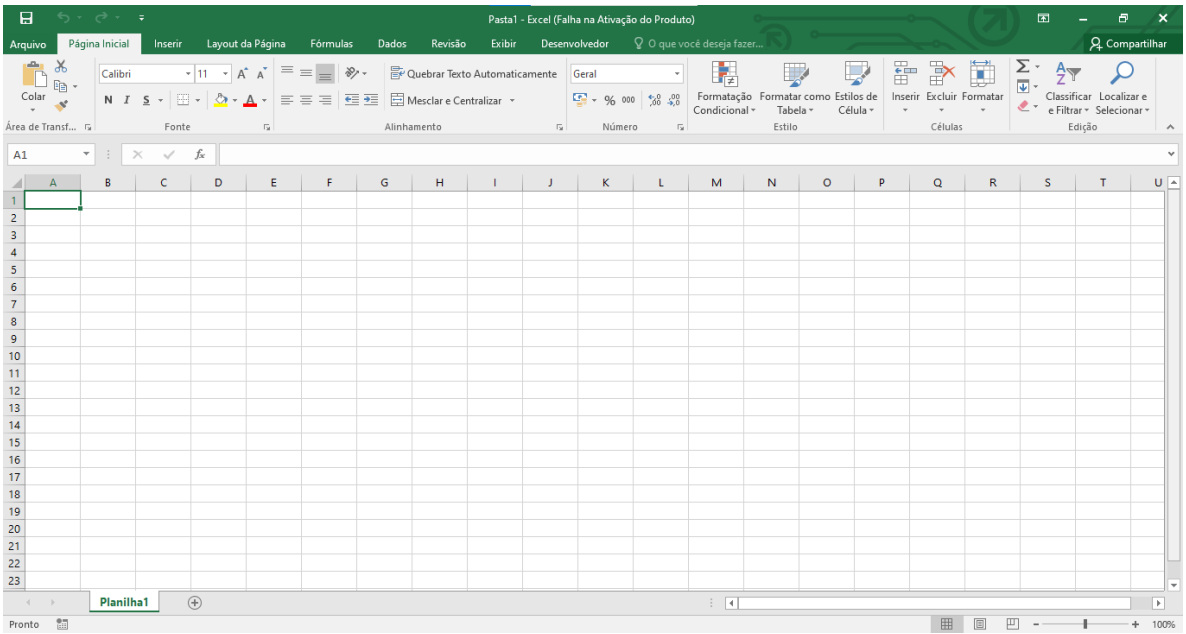
Atualmente, é comum que sistemas de automação e análise de dados utilizem SGBDs como repositórios centrais, enquanto que outras interfaces como planilhas no Excel ou no Power BI, extraiam dados para gerar relatórios dinâmicos. Essa interoperabilidade é útil, quando há necessidade de combinar grandes

volumes de dados com ferramentas conhecidas e acessíveis, como o Excel (García et al., 2020).

3.2 Microsoft Excel como ferramenta complementar de banco de dados

O Excel foi criado com o objetivo para cálculos matemáticos e financeiros, com o passar do tempo o software evoluiu para oferecer funcionalidades mais amplas, como análise de dados, automação de processos e visualização gráfica. Até o ano de 2015, o software teve um aumento significativo em sua utilização, pois ele é prático em organizar, manipular e analisar dados, possuindo uma interface acessível fazendo com que usuários sem conhecimento técnico possam manusear dados com facilidade (Walkenbach, 2015). A figura 1 ilustra a tela inicial do Excel.

Figura 1 – Planilha do Excel



Fonte: Microsoft Excel

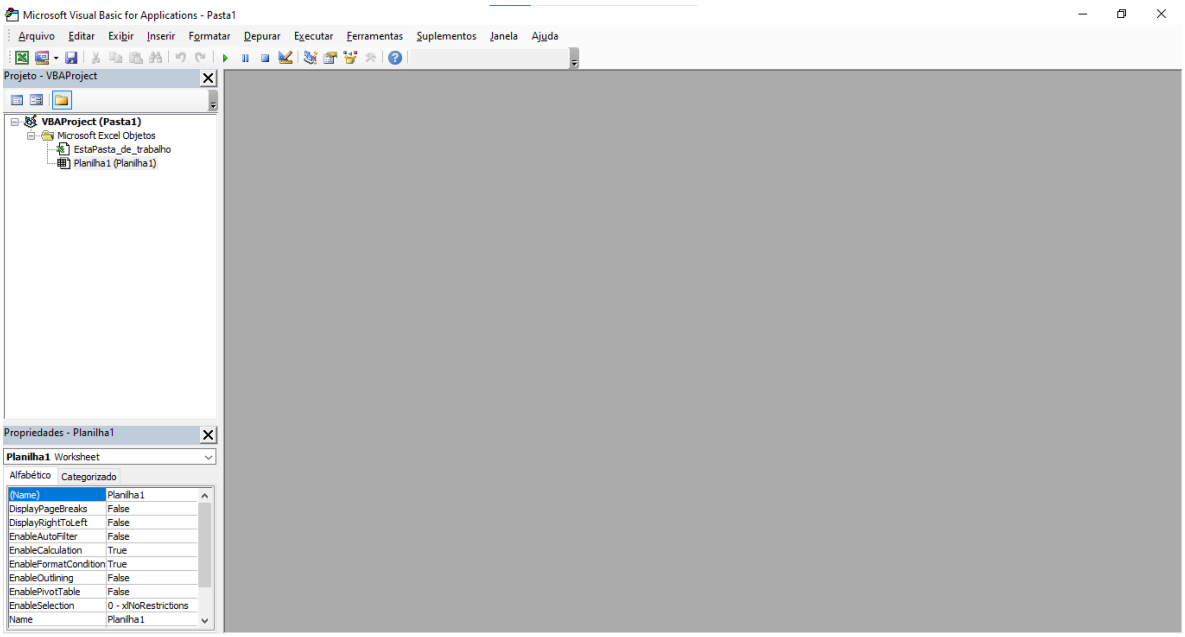
Embora não seja um sistema gerenciador de banco de dados (SGBD) tradicional, o Excel tem se mostrado uma solução viável para armazenar e gerenciar pequenas bases de dados. Por exemplo, empresas de pequeno porte frequentemente utilizam planilhas para organizar informações de clientes, estoque ou finanças, especialmente quando não há necessidade de escalabilidade ou controle transacional complexo (García et al., 2020).

O Excel também oferece ferramentas que permite o uso de linguagem SQL por meio da funcionalidade *Power Query*, ou através de conexões *Open Database Connectivity* (ODBC) traduzido como conectividade aberta de banco de dados que possibilitam a consulta às bases de dados externas diretamente na planilha. Essa integração é útil para automatizar a análise de dados em tempo real e consolidar informações de diferentes sistemas, como dados armazenados em bancos SQL podem ser importados para o Excel para gerar relatórios dinâmicos ou gráficos personalizados, facilitando a visualização de grandes volumes de dados (Chaudhuri et al., 2011).

Apesar de suas vantagens apresentadas anteriormente, o uso do Excel como banco de dados possui limitações significativas. A principal limitação está relacionada à falta de controle sobre múltiplos acessos simultâneos, o que pode causar inconsistências e erros se vários usuários tentarem editar a mesma planilha ao mesmo tempo. Além disso, o Excel possui restrições quanto ao número de linhas e colunas que pode armazenar, o que o torna inadequado para lidar com grandes volumes de dados (Walkenbach, 2015). Outro ponto que vale ressaltar é a falta de recursos de segurança robustos, como criptográfica avançada e controle de acessos granulares, que são essências em sistemas corporativos (García et al., 2020).

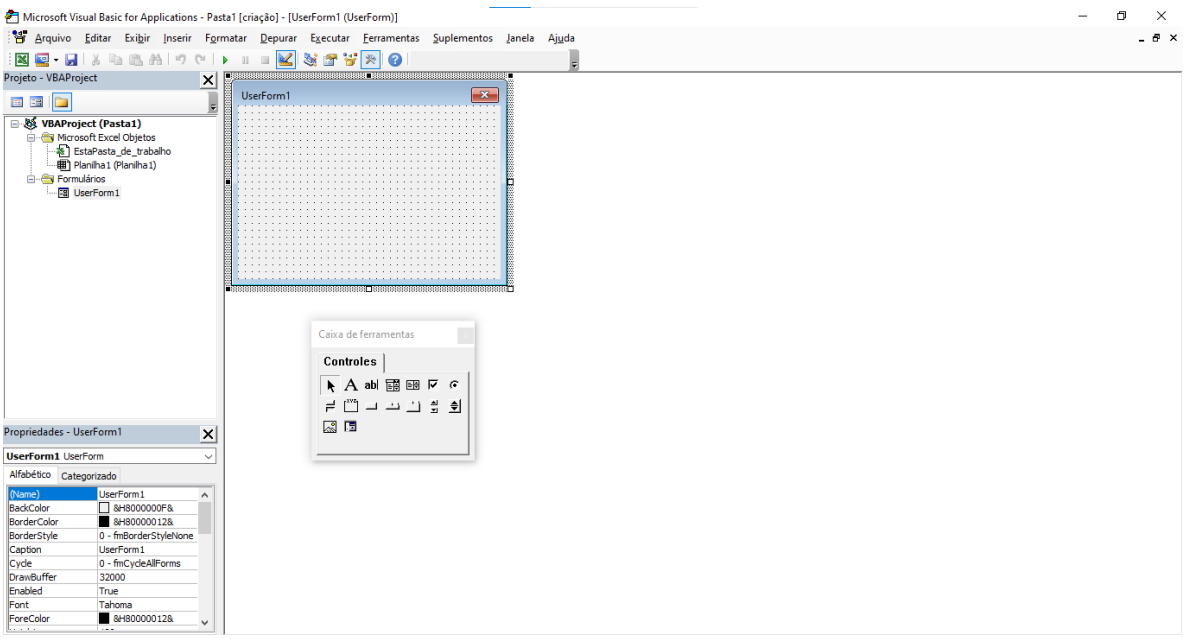
Mesmo com essas limitações, o Excel continua sendo uma ferramenta poderosa, especialmente quando usado em conjunto com outras tecnologias. Por exemplo, ao integrar o Excel com hardware como Arduino, é possível desenvolver aplicações interativas e sistemas automatizados de coletas de dados. Nesse contexto, a programação *Visual Basic for Applications* (VBA), traduzido como visual basic para aplicações, que permite que o Excel funcione não apenas como uma interface de entrada de dados, mas juntamente como um controlador de ações físicas, como acionamento de LEDs, com base em dados recebidos ou registros na planilha (Murdock, 2018). Essa versatilidade torna o Excel uma ferramenta atraente para ambientes educativos, pequenas automações, onde a simplicidade é valorizada. A interface do VBA pode ser encontrada na página inicial do Excel, clicando em desenvolvedor e por fim clicando em Visual Basic. As figuras 2 e 3 apresentam a interface do VBA.

Figura 2 – Interface do VBA



Fonte: Excel

Figura 3 – Interface do VBA para a criação do formulário



Fonte: Excel

3.3 A integração de Excel e Arduino para sistemas interativos

A combinação entre software e hardware tem ganhado destaque na automação de processos e na Internet das Coisas (IoT). Por meio da linguagem VBA do Excel, é possível programar o Excel para enviar comandos a dispositivos

físicos conectados ao Arduino, a partir do uso de bibliotecas específicas como o Microsoft Comm Control 6.0 e MSCOMSerial que possibilitam o envio e recebimento de dados por meio da porta serial, facilitando o controle de dispositivos (Souza, 2021). Essa integração exemplifica como a interatividade entre plataformas amplia as possibilidades de uso em contextos educativos e empresariais, como por exemplo monitoramento de dados em tempo real (Tech with Tim, 2020).

A utilização do Arduino permite criar sistemas reativos que respondem a dados registrados em uma planilha do Excel, como acender um LED ou acionar o sensor de som (Tech with Tim, 2020).

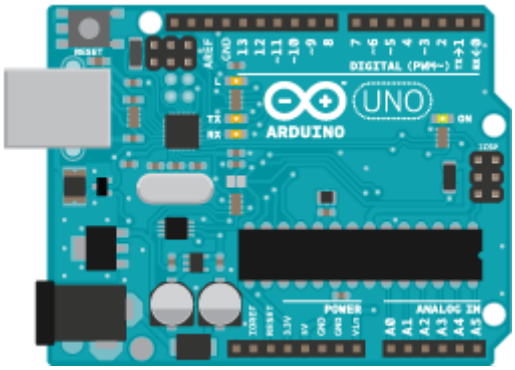
A literatura também destaca que existe a possibilidade de utilizar linguagens complementares, como o Python para intermediar a comunicação entre o Excel e dispositivos externos, ao invés da utilização de bibliotecas contidas no VBA. O uso de Python para enviar comandos ao Arduino a partir de dados armazenados no Excel é uma estratégia eficiente para expandir as capacidades do VBA, permitindo uma integração robusta e confiável (Almeida, 2019).

Implementar Python utilizando a biblioteca pySerial para automatizar a transmissão de dados entre sistemas permite que o Excel atue como interface de controle e armazenamento de dados, enquanto o Arduino executa funções físicas, como o acionamento de LEDs ou motores, com base em dados recebidos (Wang et al., 2020).

A flexibilidade do Python em manipular grandes volumes de dados e automatizar processos torna-o ideal para aplicações de controle em tempo real e sistemas de monitoramento que envolvem a coleta e resposta a dados ambientais ou sensores (Smith; Taylor, 2019). Essa combinação de tecnologias é frequentemente utilizada em contextos industriais e educacionais para criar protótipos de sistemas de automação de baixo custo, com resultados eficientes e customizáveis (González; Martinez, 2021).

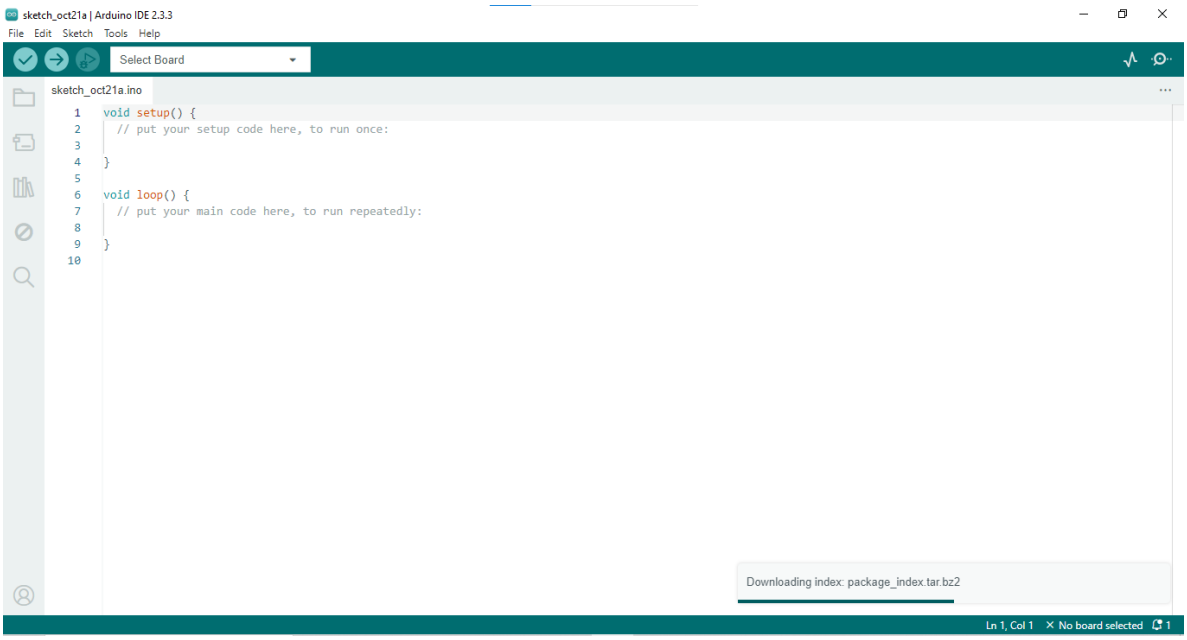
As figuras 4 e 5 apresentam o microcontrolador Arduino Uno e sua interface IDE Arduino, onde ocorre a transcrição de códigos.

Figura 4 – Dispositivo Arduino Uno



Fonte: Arduino

Figura 5 – Interface do Arduino



Fonte: Arduino

O Arduino é amplamente utilizado para automação de sistemas devido à sua simplicidade e baixo custo. Além disso, ele se destaca como uma plataforma de fácil programação e ampla compatibilidade, sendo utilizado em projetos educacionais e profissionais que exigem respostas imediatas a comandos externos (Lima e Silva, 2018).

A comunicação entre o Excel e o Arduino é estabelecida por meio de uma conexão serial, um protocolo amplamente adotado devido à sua simplicidade e confiabilidade (Fernandes; Oliveira, 2020). A taxa de transmissão (baud rate) configura-se como um dos fatores críticos para uma comunicação bem-sucedida, sendo necessário ajustar a velocidade de transmissão para garantir a sincronia

entre os dispositivos, além disso em sistemas de monitoramento e controle, o ajuste correto do baud rate previne perdas de dados e permite que o Arduino receba e processe comandos com precisão (Silva et al, 2019).

Essa comunicação serial é fundamental para o desenvolvimento de projetos de automação integrados ao Excel, onde cada dado registrado pode ser convertido em uma ação no microcontrolador, permitindo um controle eficiente e em tempo real.

3.4 Análise de casos práticos no GitHub

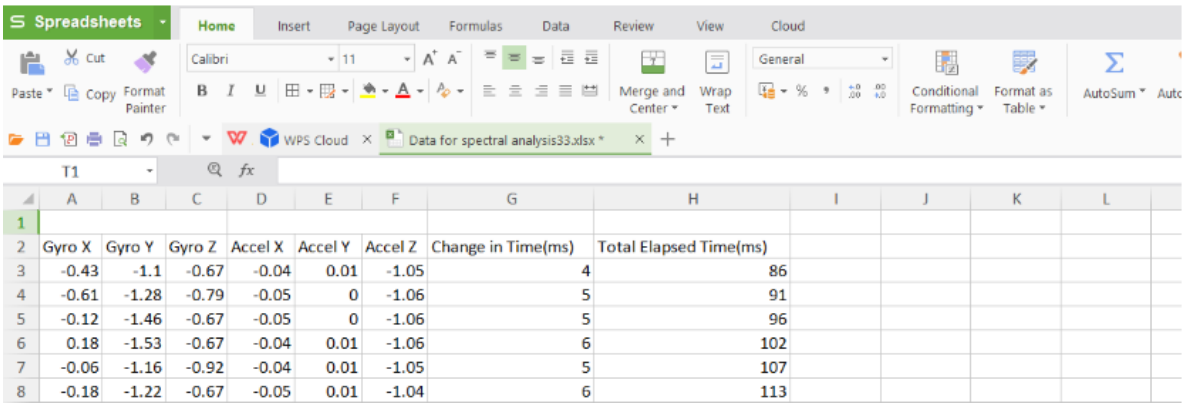
Dada a escassez de literatura científica aplicada sobre o uso do Excel e Arduino combinado, analisa-se três projetos interessantes que implementam soluções semelhantes ao tema do trabalho.

O primeiro projeto: “Data Logger with Excel Connection” elaborado pelo usuário sensorsiot permite registrar dados de um sensor no Arduino e transferi-los para o Excel automaticamente. Ele utiliza o Arduino para medir variáveis analógicas, como tensão e corrente, e conecta-se ao Excel por meio de um teclado virtual, ou seja, o Arduino emula as teclas e envia os dados diretamente para as células do Excel, sem a necessidade de softwares intermediários.

O código do projeto do usuário sensorsiot inclui uma interface de exibição no display OLED que apresenta os valores coletados em tempo real, enquanto os dados são enviados ao Excel a cada 0,5 segundo. O projeto utiliza um pino configurado para alternar entre a coleta de dados e o envio ao Excel, permitindo ao usuário controlar quando os dados são registrados. Além disso, ele realiza o ajuste de variáveis, como calibração de tensão e corrente, por meio de um botão giratório.

O segundo projeto: “Data Logging no Excel com Arduino” desenvolvido pelo usuário John Schiltz, registra dados de sensores conectados ao Arduino e salva diretamente no Excel para análise posterior, como apresenta a figura 6.

Figura 6 – Planilha do Excel recebendo os dados dos sensores conectados ao Arduino



	A	B	C	D	E	F	G	H
1								
2	Gyro X	Gyro Y	Gyro Z	Accel X	Accel Y	Accel Z	Change in Time(ms)	Total Elapsed Time(ms)
3	-0.43	-1.1	-0.67	-0.04	0.01	-1.05	4	86
4	-0.61	-1.28	-0.79	-0.05	0	-1.06	5	91
5	-0.12	-1.46	-0.67	-0.05	0	-1.06	5	96
6	0.18	-1.53	-0.67	-0.04	0.01	-1.06	6	102
7	-0.06	-1.16	-0.92	-0.04	0.01	-1.05	5	107
8	-0.18	-1.22	-0.67	-0.05	0.01	-1.04	6	113

Fonte: John Schiltz, 2024

Além disso, nesse projeto é destacado que para o Excel receber os dados dos sensores conectados ao Arduino, é necessário o nome da porta corretamente conectada no VBA para receber os dados dos sensores conectados ao Arduino, como apresentado na programação pela figura 7.

Figura 7 – Programação no VBA do Excel

```
Sub ReceberDadosArduino()
    Dim porta As String
    porta = "COM3" ' Substitua pelo nome da porta correta
    Dim linha As Integer
    linha = 1 ' Linha onde os dados serão inseridos

    ' Código para receber dados serialmente do Arduino e inserir no
    '  ...
End Sub
```

Fonte: John Schiltz, 2024

O terceiro projeto: “Data Logger with Excel connection” criado pelo usuário Sensorslot, parecido com o primeiro projeto analisado, registra valores de tensão (V) e corrente (A) armazenando essas leituras em uma planilha do Excel. O Arduino é utilizado para leitura dos dados analógicos, como a tensão e corrente, e também contém a lógica de coleta e envio dos dados para a planilha através da comunicação serial, enquanto que a planilha do Excel é conectada via script com o intuito de receber e armazenar os dados lidos. O diferencial desse projeto é a capacidade de monitorar variáveis elétricas em tempo real, possibilitando a análise e a visualização de dados de maneira dinâmica e eficiente.

Os projetos apresentados acima demonstram a viabilidade e a eficácia da integração entre Excel e Arduino para criar sistemas interativos que combinam a simplicidade de planilhas com a robustez e flexibilidade de um microcontrolador. Através da análise desses casos práticos, fica evidente que, apesar da escassez de literatura científica detalhada, há um considerável interesse e desenvolvimento de soluções práticas na comunidade de desenvolvedores, que compartilham suas experiências e conhecimentos em plataforma como GitHub.

4 MATERIAIS E MÉTODOS

4.1 Materiais

Para a integração entre a programação VBA no Excel e o microcontrolador Arduino foi realizada através de uma placa Arduino UNO, atuadores LEDs, cabo de conexão USB, *protoboard*, *jumpers*, resistências de 220 Ω, Microsoft Excel com suporte a VBA, Arduino IDE, script em Python e um computador para o desenvolvimento de códigos em VBA, Python e no IDE do Arduino.

4.2 Métodos

O método do projeto adota uma abordagem de pesquisa aplicada, pois busca desenvolver um sistema funcional e interativo com a utilização de ferramentas computacionais e eletrônicas. Além disso, o projeto proporciona uma oportunidade prática para explorar o potencial das ferramentas de automação e processamento de dados, a seguir é apresentado o desenvolvimento do projeto dividido em etapas.

4.2.1 VBA

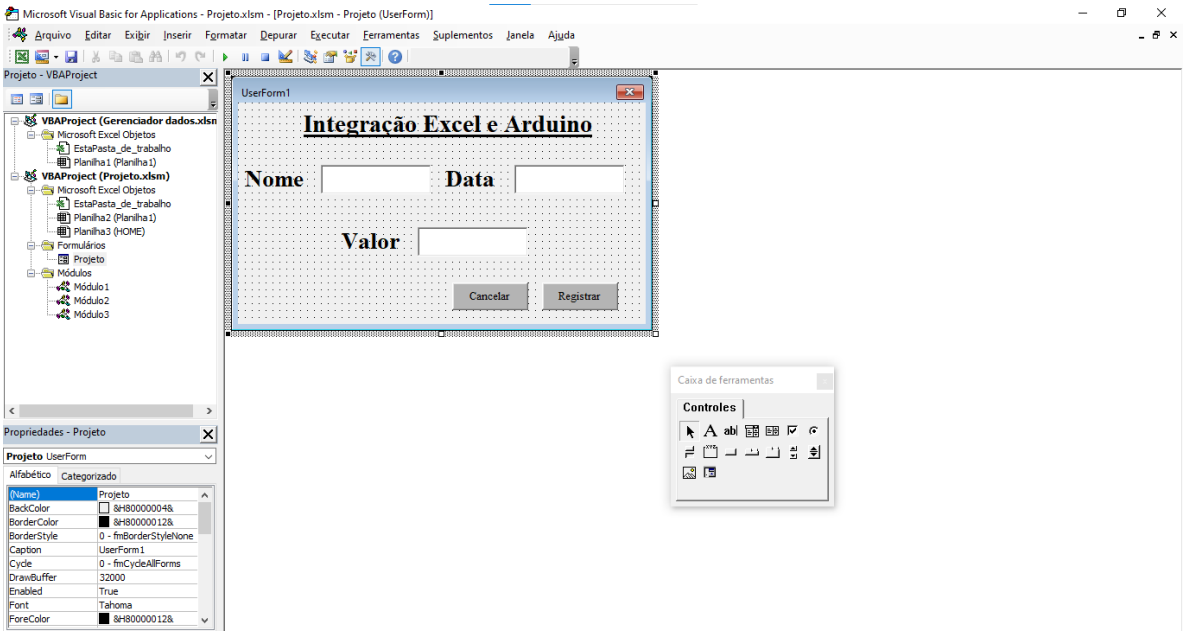
Utilizando a página inicial do Excel foi criado um botão, figura 8, para que quando o usuário clicar no botão exibe um formulário, representado pela figura 10.

Figura 8 – Botão para inserir manualmente os dados

Fonte: Autoria própria

Para a construção do formulário foi necessário utilizar o Excel VBA, onde foram inseridas caixas de texto (*Text.Box*) para o preenchimento de dados manualmente apresentada pela figura 9, em que aparece as caixas de texto do nome da pessoa e o valor a ser inserido, enquanto que a data será apresentada no formulário automaticamente, como apresenta a figura 10.

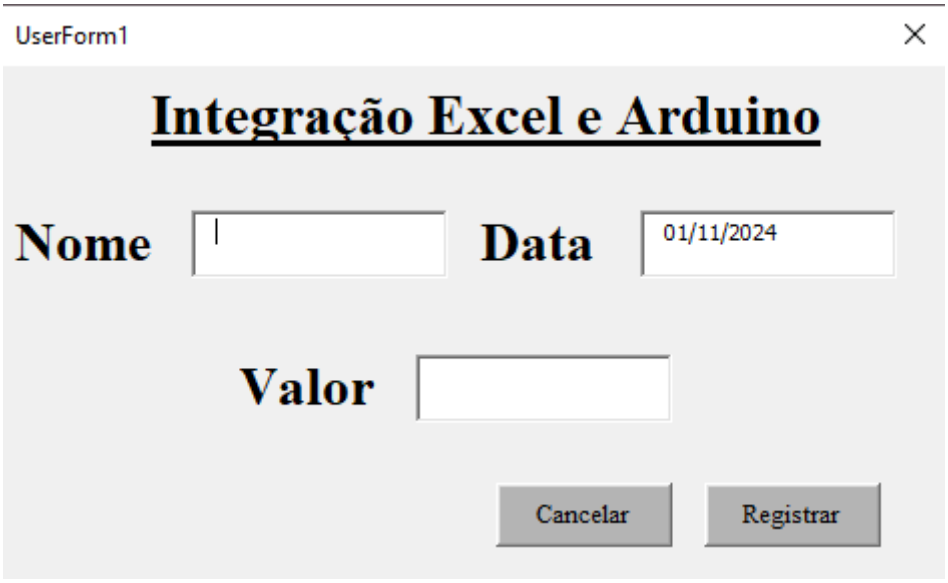
Figura 9 – Tela do VBA para a construção do formulário



Fonte: Autoria própria

A figura 10 mostra como ficou o formulário para o usuário cadastrar os dados.

Figura 10 – Interface do formulário



Fonte: Autoria própria

O código inicialização do formulário apresenta a inicialização do formulário com a data já preenchida, sem que o usuário precise adicionar a data.

Inicialização do formulário:

```
Private Sub UserForm_Initialize()
    TextBox2 = Format(Date, "dd/mm/yyyy")
    TextBox1.SetFocus
End Sub
```

O código acima utilizou o método UserForm_Initialize que é executado automaticamente quando um formulário é carregado. Dessa forma, a data já é apresentada na tela do formulário para o usuário, otimizando seu tempo.

Já a programação detalhada para a conexão entre o formulário e o botão na tela inicial está apresentada em **APÊNDICE A**, vale destacar que a interface do formulário facilita a entrada de dados, e que se os campos obrigatórios, como nome e valor, não estiverem preenchidos não é possível registrar, pois aparece uma tela exibindo uma mensagem de preenchimento do dado obrigatório.

Também ressalta-se que os valores digitados tem que estar entre 0 e 100 ou iguais a eles, ou seja, para os valores digitados no formulário que estão negativos ou acima de 100, o próprio formulário não permite o cadastro desse dado e vai aparecer uma mensagem: “O valor deve estar entre 0 e 100”, como pode ser observado pelo código verificação do valor.

Verificação do valor:

```
 ' Verifica se o valor em TextBox3 está entre 0 e 100
    Dim valorTextBox3 As Integer
    If Not IsNumeric(TextBox3.Text) Then
        MsgBox "O valor deve ser numérico."
        TextBox3.SetFocus
        Exit Sub
    End If
    valorTextBox3 = CInt(TextBox3.Text)
    If valorTextBox3 < 0 Or valorTextBox3 > 100 Then
        MsgBox "O valor deve estar entre 0 e 100."
        TextBox3.SetFocus
```

```
        Exit Sub
    End If
```

Além disso, a programação VBA permitiu a inserção, atualização, exclusão e consulta de dados no banco de dados. Por exemplo, o código campos obrigatórios no formulário verifica se os campos obrigatórios foram preenchidos antes de inserir os dados.

Campos obrigatórios no formulário:

```
    If TextBox1.Text = "" Then
        MsgBox "Obrigatório o preenchimento do nome"
        TextBox1.SetFocus
        Exit Sub
    End If
    If TextBox3.Text = "" Then
        MsgBox "Obrigatório o valor encontrado"
        TextBox3.SetFocus
        Exit Sub
    End If
```

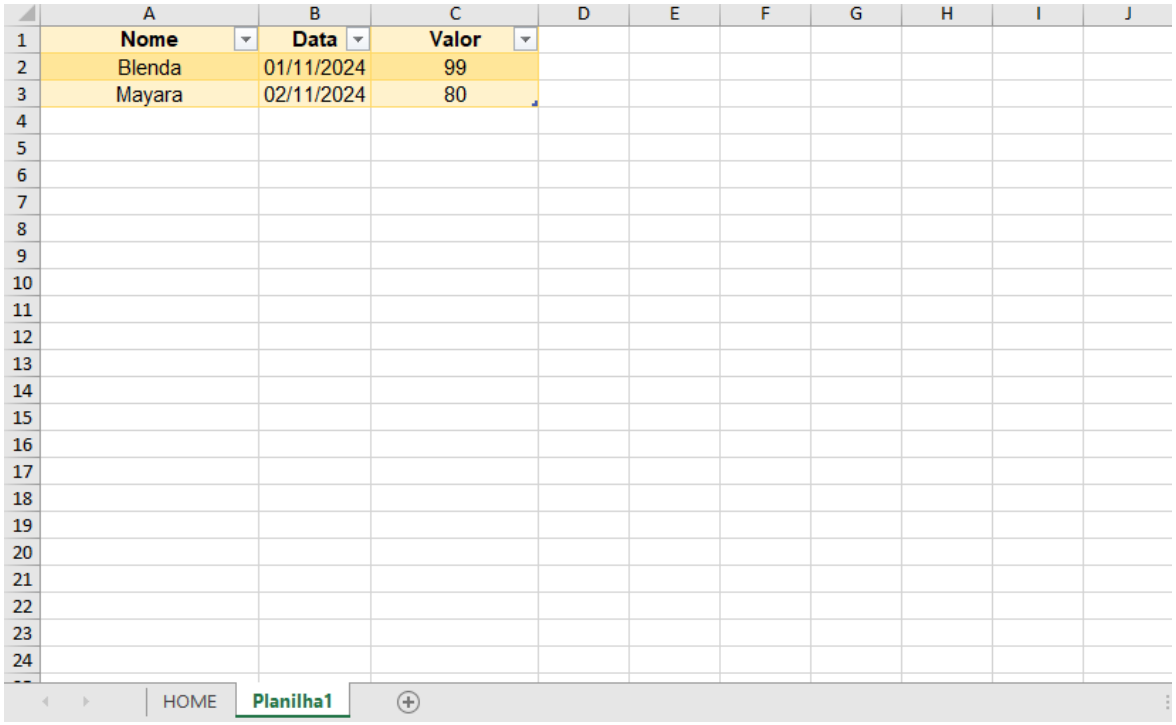
Esse trecho de código assegura que o nome e o valor são inseridos antes de prosseguir com a inserção no banco de dados. A validação de campos obrigatórios garante a integridade dos dados antes de inseri-los nas planilhas. O código VBA inclui tratamento de erros para informar ao usuário sobre quaisquer falhas e desconecta as planilhas corretamente para evitar problemas de memória.

Além disso, foi estabelecido uma conexão entre o Excel VBA e duas planilhas Excel utilizando a biblioteca ADODB. Ele configura conexões para os arquivos denominados “Projeto.xlsm” e “Gerenciador dados.xlsm”, permitindo a leitura, robustez no tratamento de dados, visto que o arquivo “Gerenciador dados.xlsm” pode ser backup dos dados do arquivo original que está contido o formulário (“Projeto.xlsm”) e também utilizar os dados contidos no “Gerenciador dados.xlsm” para elaboração de possíveis gráficos e até mesmo gráficos dinâmicos com o auxílio do Power BI.

As figuras 11 e 12 apresentam os arquivos “Projeto.xlsm” e “Gerenciador dados.xlsm”, respectivamente, e a figura 13 apresenta o local dos arquivos,

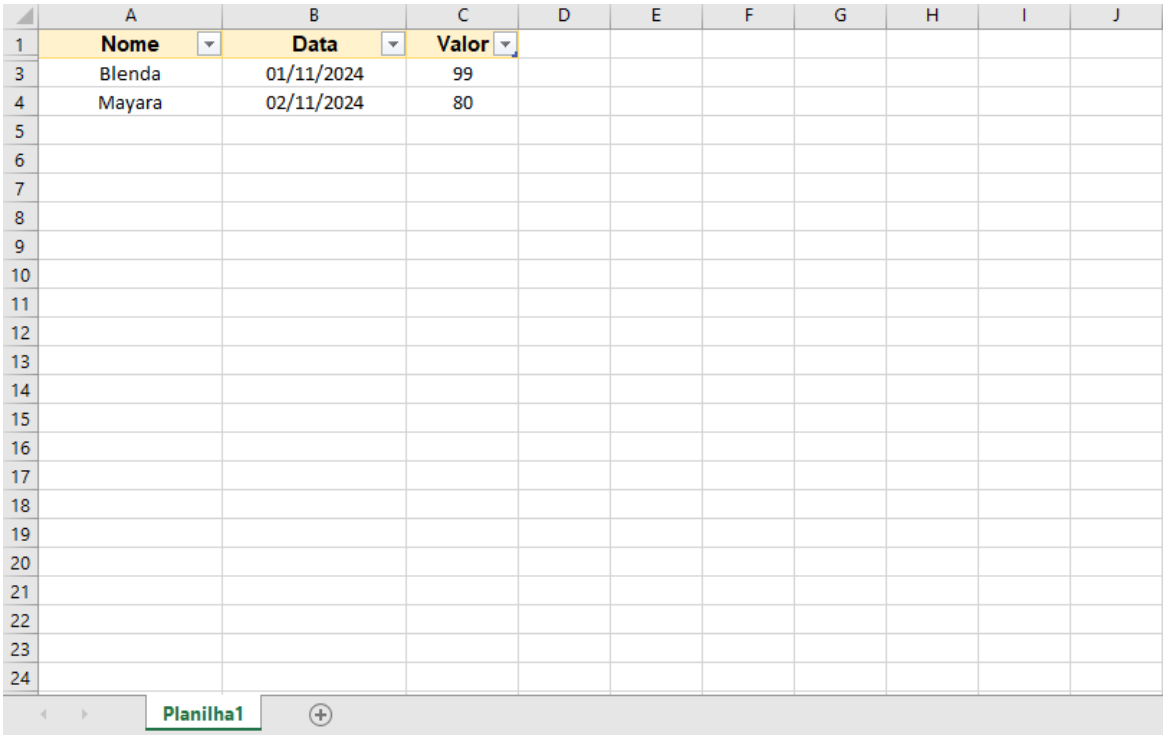
comprovando a existência de dois arquivos em Excel presentes no computador do usuário.

Figura 11 – Arquivo “Projeto.xlsm”



Fonte: Autoria própria

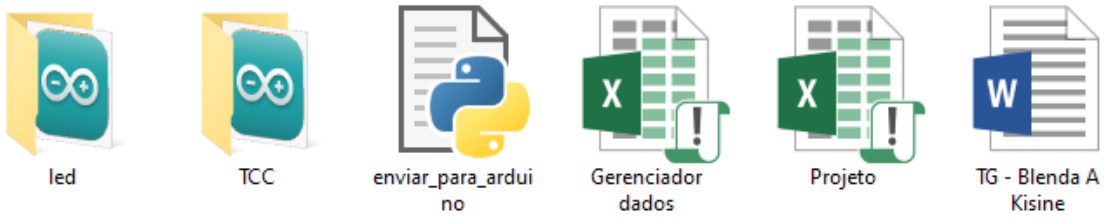
Figura 12 – Arquivo “Gerenciador dados.xlsm”



Fonte: Autoria própria

Nota-se que os dados são iguais, o que confirma que a conexão foi estabelecida com sucesso e que a transferência de dados para o arquivo “Gerenciador dados.xlsm” está ocorrendo conforme o esperado.

Figura 13 – Local dos arquivos no computador



Fonte: Autoria própria

A figura 13 apresenta a existência dos arquivos “Projeto.xlsm” e “Gerenciador dados.xlsm” localizados na pasta em que foi desenvolvido o projeto.

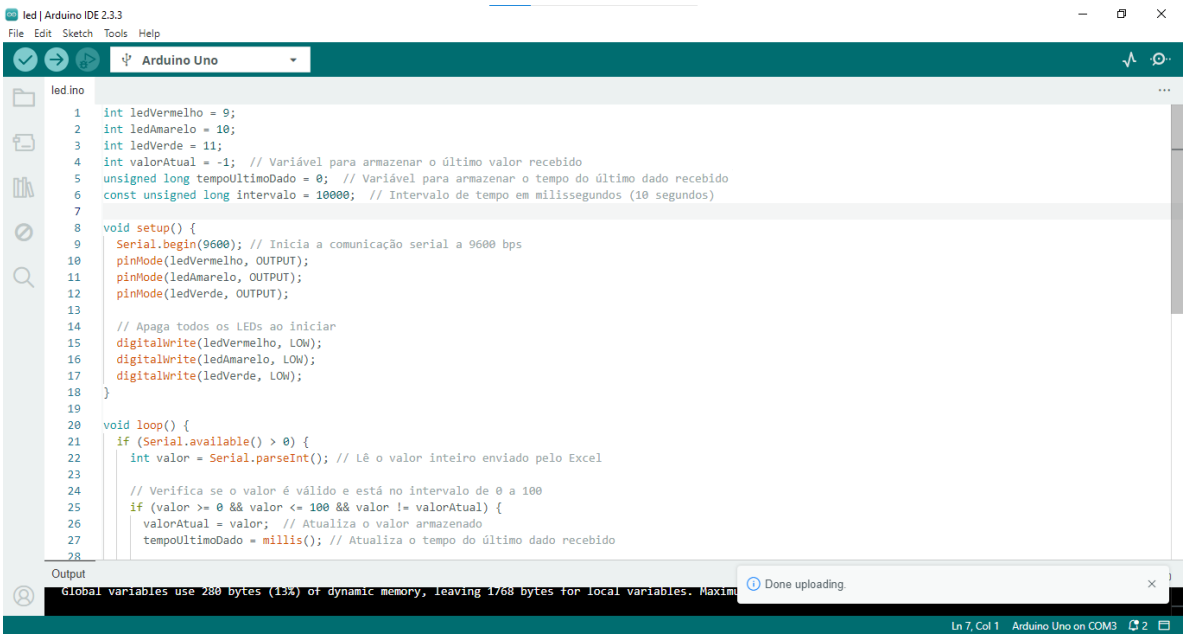
4.2.2 Arduino

No desenvolvimento deste projeto, a utilização do Arduino foi essencial para a automação do processo de controle de LEDs a partir de dados enviados pelo Excel com o auxílio do script em Python, que será abordada com detalhes posteriormente.

O Arduino foi configurado como receptor de valores transmitidos via comunicação serial, com a finalidade acionar os LEDs específicos de acordo com as faixas de valores definidos no projeto, ou seja, quando os valores forem abaixo ou igual a 40 o LED na cor vermelho acende, para valores acima de 40 e abaixo ou igual a 75 o LED na cor amarelo acende, e por fim valores acima de 75 e abaixo ou igual a 100 o LED na cor verde acende.

A figura 14 apresenta o código desenvolvido no Arduino, e também pode-se encontrar mais detalhes no **APÊNDICE C**.

Figura 14 – Código sketch desenvolvido no Arduino



Fonte: Autoria própria

Essa integração trouxe simplicidade e praticidade ao projeto, visto que, uma vez configurado, o Arduino processa e reage automaticamente nos dados, dispensando interação manual contínua, a figura 15 apresenta-se o circuito desenvolvido para conectar os LEDs ao Arduino.

No código IDE do Arduino também não é possível receber valores abaixo de 0 e acima de 100, como apresenta-se o código verificação do valor no Arduino: Verificação do valor no Arduino:

```
    if (valor >= 0 && valor <= 100 && valor != valorAtual) {
      valorAtual = valor;  // Atualiza o valor armazenado
      tempoUltimoDado = millis(); // Atualiza o tempo do último dado
recebido
```

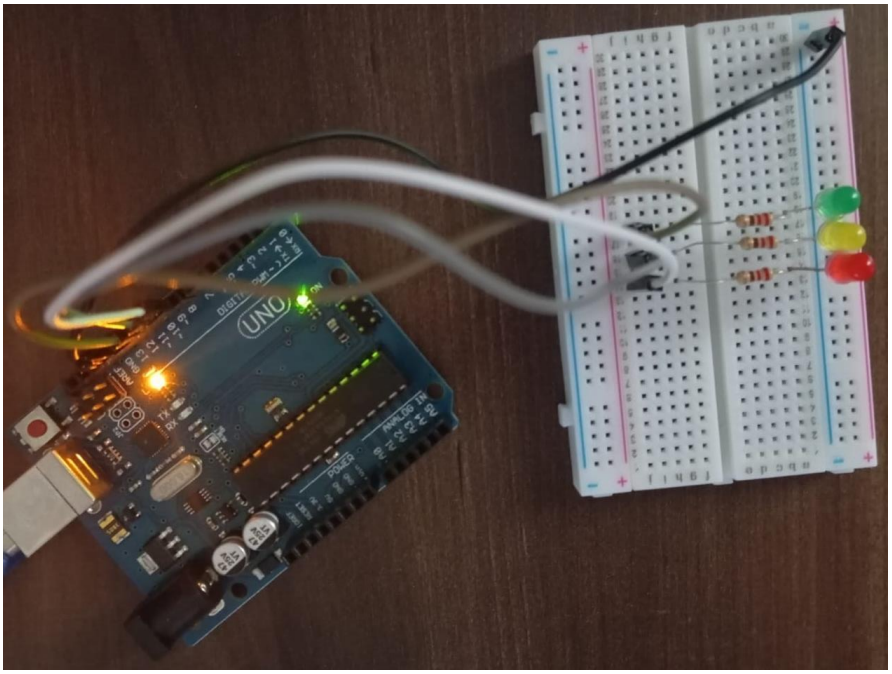
Foi realizado isso como forma de duplo check para o recebimento dos valores, então tanto no VBA quanto no IDE tem uma programação para não receber valores fora do especificado.

Destaca-se a importância da declaração dos pinos para cada LED e conectá-los de maneira correta, pois senão o comportamento do circuito será inesperado. Por exemplo, se o LED na cor vermelho estiver no pino 10, ele não acenderá quando o código tentar acender o LED no pino 9.

Também ressalta que a porta COM3 é a interface do qual o Arduino se comunica com o computador, é importante garantir que o software esteja

configurado para utilizar na porta correta, caso contrário a comunicação não funcionará, ou seja, os dados não serão enviados para o Arduino e por consequência o LED não vai ser acionado. A figura 15 apresenta-se o circuito desenvolvido para o acionamento dos LEDs.

Figura 15 – Circuito desenvolvido para o acionamento dos LEDs.



Fonte: Autoria própria

Para a comunicação entre o Excel e o Arduino, foi utilizada a biblioteca pySerial de controle de portas seriais do Python, que permitiu enviar comandos do Excel para o Arduino, pois a versão que foi utilizada para desenvolver o projeto foi a versão de 2016, então as bibliotecas Microsoft Comm Control 6.0 e MSCOMSerial não estavam contidas nele.

O código VBA abaixo apresenta que o valor do Excel envia esse valor inserido para um script Python:

```
Private Sub EnviarParaArduino(valor As Integer)
    Dim comando As String
    comando = "python ""C:\Users\User\Documents\TCC\enviar_para_arduino.py"" " & valor
    Shell comando, vbNormalFocus
End Sub
```

O VBA, linguagem utilizada no desenvolvimento do formulário e dos comandos no Excel, não possui suporte nativo para a comunicação serial com o Arduino, devido às limitações inerentes as versões atuais do VBA. Portanto, foi necessário o uso de um script em Python, apresentando no **APÊNDICE B**, para intermediar essa comunicação.

Dessa forma, o Excel ao receber o comando para enviar um valor ao Arduino, executa automaticamente o script em Python, que converte o valor em um comando serial para o microcontrolador, estabelecendo assim uma ponte entre as duas plataformas.

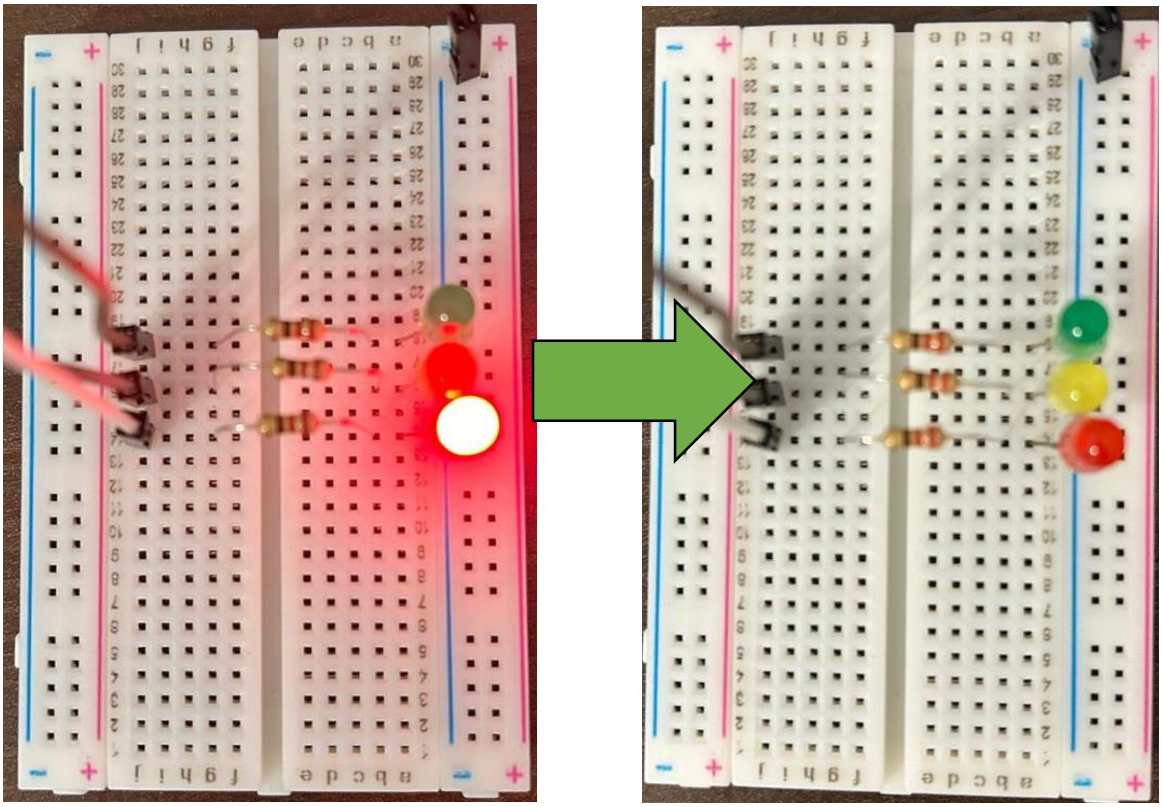
A integração entre as ferramentas foi configurada para que, sempre que o usuário inserir um valor no formulário e clicar no botão de envio, para confirmar a operação, o Excel executa o script em Python, que converte o valor em um comando serial para o Arduino.

Quando o Arduino recebe esse valor, e entende em qual faixa o valor se encontra, ele vai acionar o LED correto, depois o Arduino acende o LED vermelho, como formar de encerrar o processo, e depois de 10 segundos o LED vermelho apaga e o Arduino aguarda o próximo valor. As figuras 16, 17 e 18 apresentam os acionamentos dos LEDs vermelho, amarelo e verde.

Valores abaixo de 40:

Figura 16 – (a) Acionamento do LED na cor vermelho (b) LED apagado após 10 segundos

(a) (b)

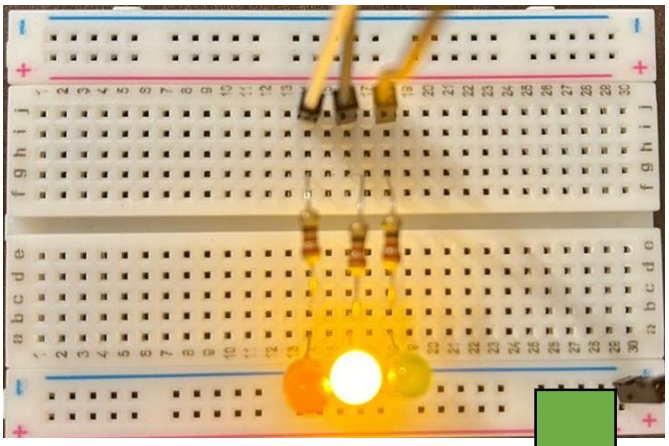


Fonte: Autoria própria

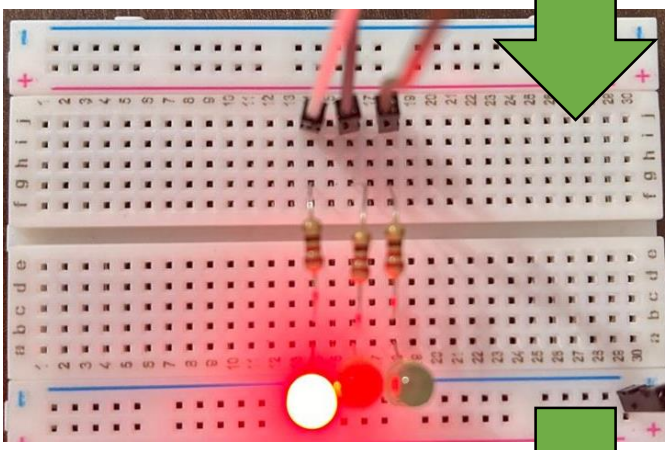
Valores acima de 40 e abaixo ou igual a 75:

Figura 17 – (a) Acionamento do LED na cor amarelo (b) LED na cor vermelho aceso (c) LED na cor vermelho apagado após 10 segundos

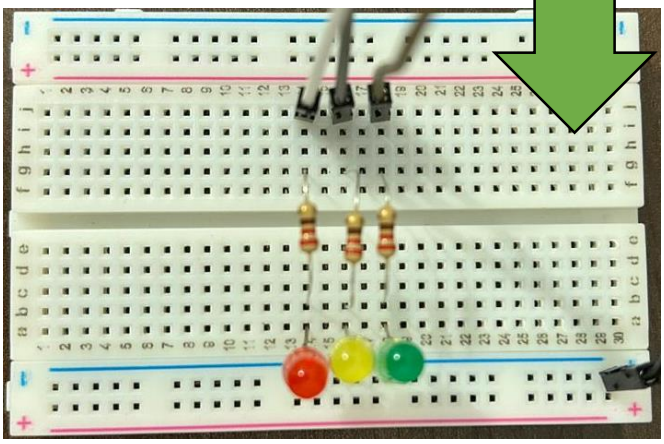
(a)



(b)



(c)

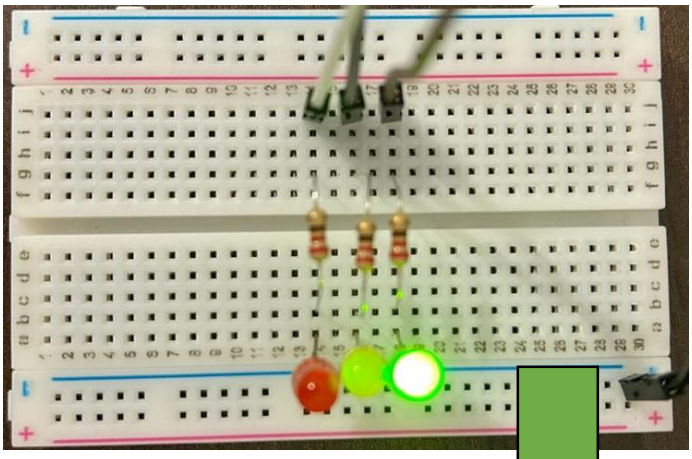


Fonte: Autoria própria

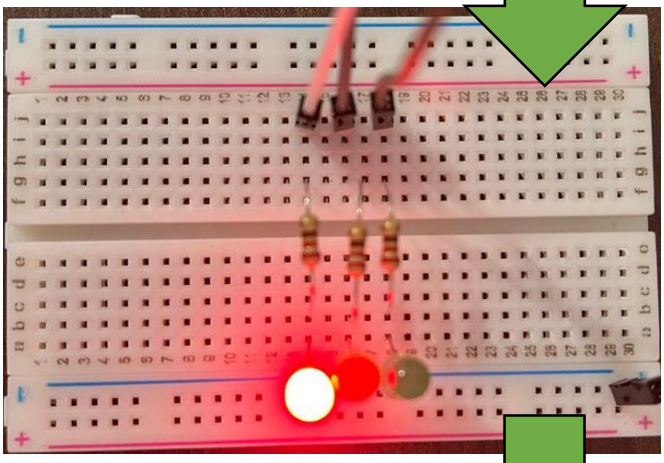
Valores acima de 75 e abaixo ou igual a 100:

Figura 18 – (a) Acionamento do LED na cor verde (b) LED na cor vermelho aceso (c) LED na cor vermelho apagado após 10 segundos

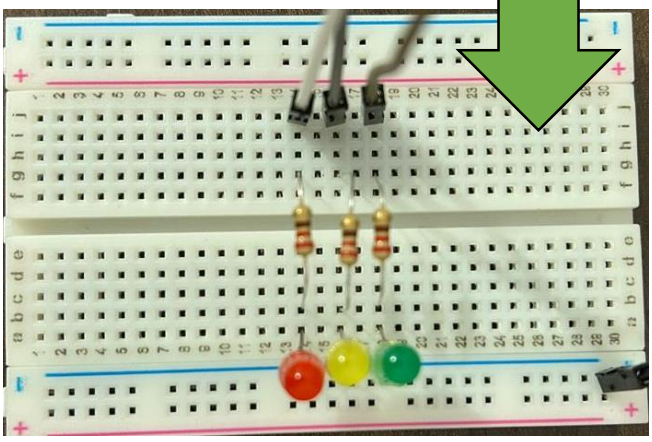
(a)



(b)



(c)



Fonte: Autoria própria

5 RESULTADOS E DISCUSSÕES

A utilização do Excel VBA para criar formulário e gerenciar dados mostrou-se uma escolha prática e eficiente. A conexão entre o Excel e duas planilhas distintas, utilizando a biblioteca ADODB, permitiu que os dados fossem replicados e armazenados de forma segura em um arquivo de backup, como o “Gerenciador dados.xlsm”, enquanto que o arquivo principal “Projeto.xlsm” era utilizado para a inserção e consulta dos dados.

Também, é possível que as duas planilhas estejam em locais diferentes, pois foram declarados o caminho de onde o arquivo está localizado, como apresenta o código caminho do diretório dos arquivos, que especifica a localização exata de um arquivo no computador.

Caminho do diretório dos arquivos:

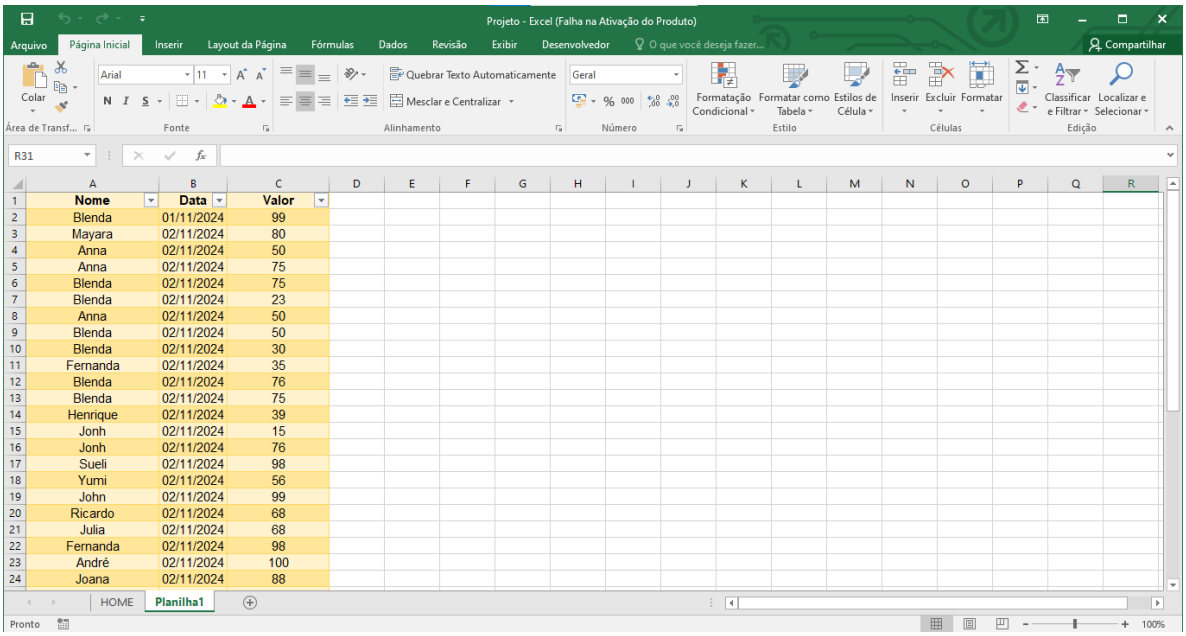
```
Dim strConexao1 As String, strConexao2 As String
    Dim provider As String, Ex As String
    Dim EnderecoPlan1 As String, EnderecoPlan2 As String
    provider = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source="
    EnderecoPlan1 = "C:\Users\User\Documents\TCC\Projeto.xlsm;"
    EnderecoPlan2 = "C:\Users\User\Documents\TCC\Gerenciador dados.xlsm;"
    Ex = "Extended Properties=""Excel 12.0 Xml;HDR=YES"";"
    ' Inicializa as conexões
    Set Conexao1 = New ADODB.Connection
    Set Conexao2 = New ADODB.Connection
```

Com isso, é possível que os arquivos possam ser movidos para diferentes pastas ou até outros dispositivos, desde que o caminho seja atualizado no código. Por exemplo, numa empresa se o arquivo principal “Projeto.xlsm” poderia estar localizado em uma pasta compartilhada acessível a todos os funcionários que precisam interagir com os dados e realizar inserções dados. Ao estar em um local comum, o arquivo principal é de fácil acesso para as operações diárias, permitindo que as pessoas possam adicionar, atualizar e consultar dados conforme necessário. Enquanto que o segundo arquivo “Gerenciador dados.xlsm” que armazena uma cópia pode ser colocado em uma pasta restrita ou em um local seguro acessível apenas por funcionários autorizados.

Dessa forma, esse arquivo fica protegido de alterações não autorizadas, e apenas funcionários autorizados, podem modificar e acessar esse arquivo. Esse tipo de situação protege os dados de alterações não intencionais e oferece uma camada adicional de segurança, minimizando o risco de usuários comuns possam adulterar ou apagar os dados de forma proposital ou acidental.

Para avaliar a eficácia da integração entre o Arduino e o Excel via VBA, foram realizados diversos testes de inserção e exclusão de dados no banco de dados, simulando o comportamento real de uso. Esses testes consistiram em adicionar valores distintos no formulário do Excel para observar se o Arduino recebia e interpretava corretamente os comandos, acionando o LED correspondente de acordo com a faixa estabelecida. Cada simulação buscava cobrir todas as condições possíveis dos LEDs, valores abaixo de 40, entre 40 e 75 e acima de 75, como apresentado pela figura 19.

Figura 19 – Inserção de valores no formulário para validação da programação



	A	B	C
1	Nome	Data	Valor
2	Blenda	01/11/2024	99
3	Mayara	02/11/2024	80
4	Anna	02/11/2024	50
5	Anna	02/11/2024	75
6	Blenda	02/11/2024	75
7	Blenda	02/11/2024	23
8	Anna	02/11/2024	50
9	Blenda	02/11/2024	50
10	Blenda	02/11/2024	30
11	Fernanda	02/11/2024	35
12	Blenda	02/11/2024	76
13	Blenda	02/11/2024	75
14	Henrique	02/11/2024	39
15	Jonh	02/11/2024	15
16	Jonh	02/11/2024	76
17	Sueli	02/11/2024	98
18	Yumi	02/11/2024	56
19	John	02/11/2024	99
20	Ricardo	02/11/2024	68
21	Julia	02/11/2024	68
22	Fernanda	02/11/2024	98
23	André	02/11/2024	100
24	Joana	02/11/2024	88

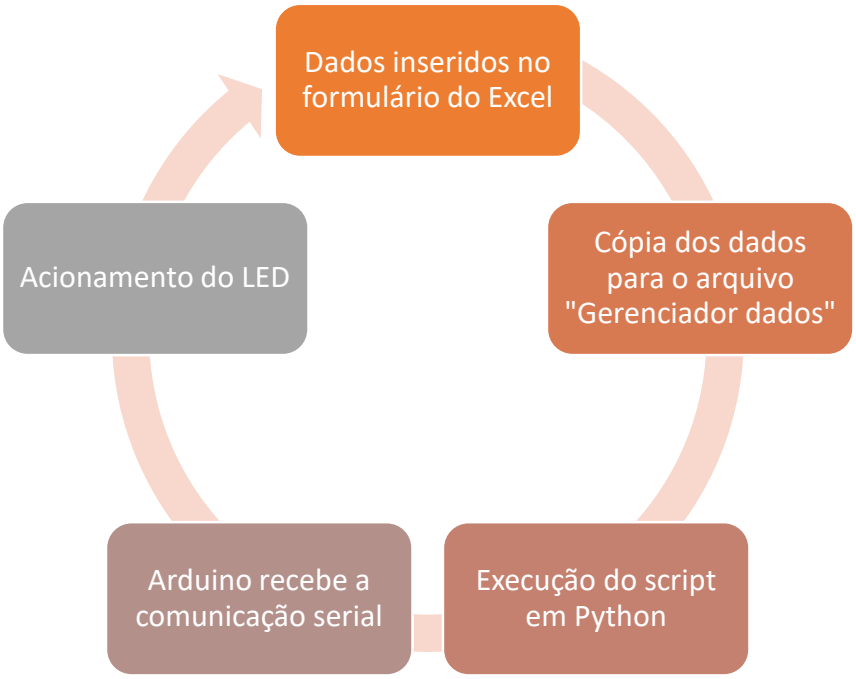
Fonte: Autoria própria.

O uso do Arduino, aliado ao controle via VBA e à comunicação serial intermediada pelo Python, possibilitou a criação de um sistema de automação eficaz e intuitivo. Mas para as versões do Excel que possuem as bibliotecas Microsoft Comm Control 6.0 e MSCOMSerial, essas bibliotecas fariam a função da comunicação serial intermediada pelo script em Python.

A cada entrada no banco de dados do Excel, o Arduino recebia e interpretava o valor, proporcionando uma resposta visual imediata e visível através dos LEDs, conforme as figuras 16, 17 e 18. Os critérios de validação envolveram verificar visualmente o acionamento correto dos LEDs. Essa validação foi crucial para confirmar que o LED acionado refletia corretamente o valor atual armazenado no banco de dados, promovendo um feedback visual imediato.

Como resultado, os testes de repetição demonstraram que o sistema é estável, respondendo rapidamente às novas inserções de valores. Esses resultados mostram que a integração entre Arduino, o Excel e o VBA são capazes de realizar controle direto de componentes eletrônicos com um tempo de resposta imediato e com baixo custo de implementação. A figura 19 apresenta-se um fluxograma para retratar o fluxo do processo como um todo, desde a inserção de dados até o acionamento do LED.

Figura 19 – Fluxograma do processo do projeto



Fonte: Autoria própria

Alguns desafios de comunicação foram encontrados, como ajustes na velocidade de comunicação (baud rate) no Arduino para garantir a sincronização correta entre os dispositivos. Após ajustar a taxa de transmissão para 9600 bps e garantir que ambos os dispositivos estavam sincronizados, a comunicação fluiu sem problemas.

Além disso, a utilização do Excel VBA como ferramenta para criar um banco de dados básico se mostrou uma alternativa viável e prática, especialmente para projetos de automação que não demandam grandes volumes de dados ou alta complexidade nas operações de consulta e atualização, pois o Excel não é um banco de dados tradicional. Por meio das funcionalidades de planilhas e dos comandos VBA, foi possível estruturar uma base de dados que permite a inserção, atualização, exclusão e consulta de registros, oferecendo ao usuário uma interface acessível e interativa.

O uso do Excel VBA como banco de dados pode ser considerado válido em contextos de projetos de pequena complexidade e para demonstrar princípios básicos de armazenamento e manipulação de dados, pois ele pode manipular e consultar dados de maneiras que se assemelham ao uso de SQL, por exemplo a ferramenta *Power Query* do Excel que permite transformar e analisar dados, conseguindo realizar operações de consulta e manipulação de dados. Essa abordagem pode ser especialmente útil em cursos ou projetos introdutórios, onde o objetivo é integrar ferramentas de automação e análise de dados de forma prática e acessível.

No entanto, vale ressaltar que para sistemas que demandam escalabilidade, segurança ou suporte a transações complexas, o Excel possui limitações que o torna inadequado como um substituto de banco de dados dedicados, como SQL Server ou MySQL. Portanto, embora limitado em termos de capacidade e desempenho, o uso do Excel VBA para criar um banco de dados simples é uma escolha válida para demonstrar conceitos de manipulação de dados e integração entre software e hardware, como o proposto deste projeto com o Arduino.

6 CONCLUSÃO

Este trabalho apresentou o desenvolvimento de um sistema integrado que utiliza o Microsoft Excel com VBA e o microcontrolador Arduino, aliada ao script em Python, demonstrando uma abordagem prática e eficaz para automação de processos. Através deste trabalho, foi possível identificar as vantagens e limitações de cada ferramenta, bem como a importância para o desenvolvimento deste projeto.

A pesquisa aplicada permitiu explorar a interação entre ferramentas computacionais e eletrônicas, resultando em uma interface amigável para o usuário, que possibilita a entrada e a manipulação de dados de forma intuitiva.

Através da implementação de um formulário em Excel, o usuário podia inserir dados que, ao serem validados, eram transmitidos para o Arduino via comunicação serial mediada por um script em Python. Essa configuração possibilitou o acionamento dos LEDs de acordo com as faixas de valores definidos no projeto e abordados anteriormente, demonstrando a eficiência da integração entre as plataformas.

Os testes realizados confirmaram a eficácia do sistema como inserção e exclusão de dados no Excel, também foram realizados testes para garantir que os LEDs respondessem corretamente aos valores inseridos, cobrindo todas as condições possíveis. Os resultados confirmaram a estabilidade e a funcionalidade do sistema, com respostas rápidas e precisas às novas inserções de valores.

A importância de manter duas planilhas conectadas, reside na garantia de que os dados estão sempre atualizados e disponíveis, oferecendo uma camada adicional de segurança. Essa duplicidade não só protege contra a perda de dados, mas também facilita a análise e a visualização de informações.

É extremamente importante destacar que algumas limitações foram identificadas, como a necessidade de ajustes na taxa de comunicação serial e a limitações em cenários que exige maior escalabilidade, segurança, complexidade nas operações e soluções mais robustas, o Excel como banco de dados não seria adequado para essa situação. No entanto, estas limitações não comprometem o objetivo do sistema proposto, mas sugerem áreas para melhorias e otimizações futuras.

Em suma, este projeto evidencia a eficácia da integração entre o Excel e o Arduino, e também serviu como um recurso educacional para introduzir conceitos de automação e processamento de dados. Futuras pesquisas aplicadas podem explorar o aprimoramento desta abordagem, considerando a implementação de tecnologias mais avançadas, visando atender as demandas mais complexas em automação e controle de dados.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, J. **Integração entre Excel e Arduino utilizando Python para automação de processos**. Revista de Tecnologia Aplicada, v. 5, n. 2, p. 45-58, 2019.

AL SWEIGART, A. **Automate the Boring Stuff with Python: Practical Programming for Total Beginners**. San Francisco: No Starch Press, 2015.

BANZI, M.; SHILOH, M. **Getting Started with Arduino**. 3. ed. Sebastopol: O'Reilly Media, 2014.

CHAUDHURI, S.; DAYAL, U.; **NARASAYYA, V. An Overview of Business Intelligence Technology**. Communications of the ACM, v. 54, n. 8, p. 88-98, 2011.

FERNANDES, R.; OLIVEIRA, L. **Comunicação serial e suas aplicações em projetos de automação com Arduino**. Revista de Engenharia Eletrônica, v. 11, n. 1, p. 12-25, 2020.

GARCÍA, S.; LUENGO, J.; HERRERA, F. **Data Preprocessing in Data Mining.** Springer, 2020.

GONZÁLEZ, P.; MARTINEZ, A. **Using Python and Arduino for Low-Cost Automation and Control**. Journal of Engineering and Technology Education, 2021.

LIMA, A.; SILVA, P. **Aplicações de microcontroladores em projetos de automação educacional**. Revista Brasileira de Ensino de Tecnologia, v. 7, n. 3, p. 32-40, 2018.

MACHADO, Felipe Nery Rodrigues. **Projeto e implementação de banco de dados**. 3. ed. São Paulo: Érica, 2014.

MONK, S. **Programming Arduino: Getting Started with Sketches**. 2. ed. New York: McGraw-Hill Education, 2013.

MURDOCK, J. **Programming Arduino: Getting Started with Sketches**. McGraw-Hill, 2018.

PETROV, P.; et al. **Database Administration Practical Aspects in Providing Digitalization of Educational Services**. International Journal of Emerging Technologies in Learning (Online), Vienna, v. 17, n. 20, p. 274-282, 2022.

SENSORSLOT. **Data Logger with Excel connection**. Disponível em: https://github.com/SensorsIot/Data-Logger-with-Excel-connection. Acesso em: 20 out. 2024.

SCHILTZ, John. **Arduino-Data-Logger: Easy to use set of programs that saves data from an Arduino to a spreadsheet**. Disponível em: https://github.com/schiltz3/Arduino-Data-Logger. Acesso em: 5 nov. 2024.

SENSORSIOT. **Data Logger with Excel connection**. Disponível em: https://github.com/SensorsIot/Data-Logger-with-Excel-connection. Acesso em: 1 out. 2020.

SILVA, R.; et al. **Considerações sobre a configuração de baud rate em sistemas de controle de dados**. Revista de Automação e Sistemas, v. 6, n. 2, p. 66-74, 2019.

SMITH, R.; TAYLOR, L. **Python for Hardware Control: An Integration with Microcontroller Projects**. Springer, 2019.

SOUZA, T. **Automação de sistemas com Excel VBA: possibilidades e limitações. Revista de Informática Aplicada**, v. 10, n. 1, p. 95-110, 2021.

Tech With Tim. **How to Control an Arduino with Excel**. YouTube, 2020. Disponível em: https://www.youtube.com/watch?v=V8PHw3e-aPo. Acesso em: 29 set. 2024.

VANDERPLAS, J. **Python Data Science Handbook: Essential Tools for Working with Data**. Sebastopol: O'Reilly Media, 2016.

WALKENBACH, J. **Excel 2016 Power Programming with VBA**. Wiley, 2015.

WANG, J.; LI, F.; ZHANG, X. **Integration of Python and Arduino in Control Systems Applications**. International Journal of Engineering Education, 2020.

APÊNDICE A – Código completo desenvolvido por meio do VBA do Excel (a) Formulário (b) Módulo 1 (c) Módulo 2 (d) Módulo 3

(a) Formulário

```
Private Sub CommandButton1_Click()
    Application.ScreenUpdating = False
    On Error GoTo Erro

    Dim rs As ADODB.Recordset

    ' Verifica os campos obrigatórios
    If TextBox1.Text = "" Then
        MsgBox "Obrigatório o preenchimento do nome"
        TextBox1.SetFocus
        Exit Sub
    End If
    If TextBox3.Text = "" Then
        MsgBox "Obrigatório o valor encontrado"
        TextBox3.SetFocus
        Exit Sub
    End If

    ' Verifica se o valor em TextBox3 está entre 0 e 100
    Dim valorTextBox3 As Integer
    If Not IsNumeric(TextBox3.Text) Then
        MsgBox "O valor deve ser numérico."
        TextBox3.SetFocus
        Exit Sub
    End If

    valorTextBox3 = CInt(TextBox3.Text)
    If valorTextBox3 < 0 Or valorTextBox3 > 100 Then
        MsgBox "O valor deve estar entre 0 e 100."
```

```
        TextBox3.SetFocus
        Exit Sub
    End If

    ' Conectar ao Projeto (Conexao1) e Gerenciador Dados (Conexao2)
    Módulo1.ConectarPlan
    Set rs = New ADODB.Recordset

    ' Inserção em Projeto.xlsm
    rs.Open "SELECT * FROM [Planilha1$]", Conexao1, adOpenKeyset,
adLockPessimistic
    rs.AddNew
    rs!Nome = TextBox1.Text
    rs!Data = TextBox2.Text
    rs!valor = TextBox3.Text
    rs.Update
    rs.Close

    ' Inserção em Gerenciador dados.xlsm
    rs.Open "SELECT * FROM [Planilha1$]", Conexao2, adOpenKeyset,
adLockPessimistic
    rs.AddNew
    rs!Nome = TextBox1.Text
    rs!Data = TextBox2.Text
    rs!valor = TextBox3.Text
    rs.Update
    rs.Close

    ' Enviar valor ao Arduino para controlar o LED
    Dim valor As Integer
    valor = CInt(TextBox3.Text)
    EnviarParaArduino valor
```

```
    ' Finalização e Limpeza
    Módulo1.DesconectarPlan
    Set rs = Nothing
    Call Limpar
    TextBox1.SetFocus
    MsgBox "Dados cadastrados com êxito."

    Application.ScreenUpdating = True
    Exit Sub

Erro:
    MsgBox "Erro: " & Err.Description, vbCritical, "ERRO"
    Application.ScreenUpdating = True
End Sub
Private Sub Limpar()
    TextBox1.Text = ""
    TextBox3.Text = ""
End Sub
Private Sub CommandButton2_Click()
    Unload Projeto
End Sub
Private Sub UserForm_Initialize()
    TextBox2 = Format(Date, "dd/mm/yyyy")
    TextBox1.SetFocus
End Sub
Private Sub EnviarParaArduino(valor As Integer)
    Dim comando As String
    comando = "python ""C:\Users\User\Documents\TCC\enviar_para_arduino.py"" "
& valor
    Shell comando, vbNormalFocus
End Sub
```

Explicando o código APÊNDICE A - (a) Formulário:

- Application.ScreenUpdating = False: desativa a atualização da tela para melhorar o desempenho durante a execução do código.
- On Error GoTo Erro: Define um manipulador de erros que redireciona a execução para o rótulo Erro se um erro ocorrer.
- Dim rs As ADODB.Recordset: Declara uma variável rs do tipo ADODB.Recordset para manipular registros de banco de dados.
- If TextBox1.Text = " " Then ... End If: Verifica se TextBox1 está vazio. Se estiver, exibe uma mensagem e define o foco no TextBox1, em seguida sai da sub-rotina. Assim como para o TextBox3.Text
- Dim valorTextBox3 As Integer: Declara uma variável valorTextBox3 do tipo inteiro.
- If Not IsNumeric(TextBox3.Text) Then ... End If: Verifica se o valor em TextBox3 é numérico. Se não for, exibe uma mensagem e define o foco no TextBox3, em seguida sai da sub-rotina.
- valorTextBox3 = CInt(TextBox3.Text): Converte o valor de TextBox3 para um inteiro e o armazena em valorTextBox3.
- If valorTextBox3 < 0 Or valorTextBox3 > 100 Then ... End If: Verifica se valorTextBox3 está entre 0 e 100. Se não estiver, exibe uma mensagem e define o foco no TextBox3, em seguida sai da sub-rotina.
- Módulo1.ConectarPlan: Chama um procedimento em Módulo1 para conectar às planilhas.
- Set rs = New ADODB.Recordset: Inicializa a variável rs como um novo Recordset.
- rs.Open "SELECT * FROM [Planilha1$]", Conexao1, adOpenKeyset, adLockPessimistic: Abre um Recordset com todos os registros da Planilha1 na conexão Conexao1.
- rs.AddNew ... rs.Update: Adiciona um novo registro com os dados dos TextBoxes e o atualiza no Recordset.
- rs.Close: Fecha o Recordset.

- rs.Open "SELECT * FROM [Planilha1$]", Conexao2, adOpenKeyset, adLockPessimistic: Abre um Recordset com todos os registros da Planilha1 na conexão Conexao2.
- rs.AddNew ... rs.Update: Adiciona um novo registro com os dados dos TextBoxes e o atualiza no Recordset.
- rs.Close: Fecha o Recordset.
- Dim valor As Integer ... EnviarParaArduino valor: Converte o valor de TextBox3 para inteiro e chama a sub-rotina EnviarParaArduino com este valor.
- Módulo1.DesconectarPlan: Chama um procedimento em Módulo1 para desconectar as planilhas.
- Set rs = Nothing: Libera a variável rs.
- Call Limpar: Chama a sub-rotina Limpar para limpar os campos do formulário.
- TextBox1.SetFocus: Define o foco no TextBox1, então o cursor vai estar no TextBox1.
- MsgBox "Dados cadastrados com êxito.": Exibe uma mensagem informando que os dados foram cadastrados com sucesso.
- Application.ScreenUpdating = True: Reativa a atualização da tela.
- Erro:: Rótulo de erro.
- MsgBox "Erro: " & Err.Description, vbCritical, "ERRO": Exibe uma mensagem de erro com a descrição do erro ocorrido.
- Application.ScreenUpdating = True: Reativa a atualização da tela.
- TextBox1.Text = " " **e** TextBox3.Text = " ": Limpa os valores de TextBox1 e TextBox3.
- Unload Projeto: Fecha o formulário chamado Projeto.
- TextBox2 = Format(Date, "dd/mm/yyyy"): Formata a data atual como "dd/mm/yyyy" e a exibe em TextBox2.
- TextBox1.SetFocus: Define o foco no TextBox1 quando o formulário é inicializado.
- Dim comando As String: Declara uma variável comando do tipo String.

- comando = "python ""C:\Users\User\Documents\TCC\enviar_para_arduino.py"" " & valor: Constrói um comando que executa um script Python, passando o valor recebido.
- Shell comando, vbNormalFocus: Executa o comando no shell do sistema com foco normal.

(b) Módulo 1

```
Public Conexao1 As ADODB.Connection
Public Conexao2 As ADODB.Connection

Sub ConectarPlan()

    On Error GoTo Erro

    Dim strConexao1 As String, strConexao2 As String
    Dim provider As String, Ex As String
    Dim EnderecoPlan1 As String, EnderecoPlan2 As String

    provider = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source="
    EnderecoPlan1 = "C:\Users\User\Documents\TCC\Projeto.xlsm;"
    EnderecoPlan2 = "C:\Users\User\Documents\TCC\Gerenciador dados.xlsm;"
    Ex = "Extended Properties=""Excel 12.0 Xml;HDR=YES"";"

    ' Inicializa as conexões
    Set Conexao1 = New ADODB.Connection
    Set Conexao2 = New ADODB.Connection

    ' Conexão com Projeto.xlsm
    strConexao1 = provider & EnderecoPlan1 & Ex
    Conexao1.Open strConexao1
```

```
    ' Conexão com Gerenciador dados.xlsm
    strConexao2 = provider & EnderecoPlan2 & Ex
    Conexao2.Open strConexao2

    Exit Sub

Erro:
    MsgBox "Erro na conexão com o banco de dados: " & Err.Description, vbCritical,
"Erro de Conexão"

End Sub
Sub DesconectarPlan()
    If Not Conexao1 Is Nothing Then
        Conexao1.Close
        Set Conexao1 = Nothing
    End If

    If Not Conexao2 Is Nothing Then
        Conexao2.Close
        Set Conexao2 = Nothing
    End If
End Sub
Sub Cadastrar()
    On Error GoTo Erro
    UserForm1.Show
    Exit Sub

Erro:
    MsgBox "Erro ao abrir o formulário!", vbCritical, "Erro no Cadastro"
End Sub
```

Explicando o código do APÊNDICE A - (b) Módulo 1

- Public Conexao1 e Public Conexao2: Declara duas variáveis públicas do tipo ADODB.Connection, que representam as conexões com os arquivos Excel. Como são públicas, essas variáveis podem ser acessadas por qualquer módulo no projeto VBA.
- On Error GoTo Erro: Define um manipulador de erros que redireciona a execução para o rótulo Erro se ocorrer um erro.
- strConexao1 e strConexao2: Strings para armazenar as cadeias de conexão.
- provider: String para o provedor OLEDB.
- EnderecoPlan1 e EnderecoPlan2: Strings para os caminhos dos arquivos Excel.
- provider: Define o provedor OLEDB a ser usado.
- EnderecoPlan1 e EnderecoPlan2: Define os caminhos completos dos arquivos Excel.
- Set Conexao1 = New ADODB.Connection: Inicializa Conexao1 como uma nova conexão ADODB. Aplica-se também Set Conexao 2 para essa justificativa.
- strConexao1 = provider & EnderecoPlan1 & Ex: Constrói a string de conexão para o primeiro arquivo Excel.
- Conexao1.Open strConexao1: Abre a conexão com Projeto.xlsm.
- strConexao2 = provider & EnderecoPlan2 & Ex: Constrói a string de conexão para o segundo arquivo Excel.
- Conexao2.Open strConexao2: Abre a conexão com Gerenciador dados.xlsm.
- Erro:: Rótulo de erro.
- MsgBox "Erro na conexão com o banco de dados: " & Err.Description, vbCritical, "Erro de Conexão": Exibe uma mensagem de erro com a descrição do erro ocorrido.
- If Not Conexao1 Is Nothing Then ... End If: Verifica se Conexao1 está inicializada.
- Conexao1.Close: Fecha a conexão Conexao1.
- Set Conexao1 = Nothing: Libera a variável Conexao1.

- If Not Conexao2 Is Nothing Then ... End If: Verifica se Conexao2 está inicializada.
- Conexao2.Close: Fecha a conexão Conexao2.
- Set Conexao2 = Nothing: Libera a variável Conexao2.
- On Error GoTo Erro: Define um manipulador de erros que redireciona a execução para o rótulo Erro se ocorrer um erro.
- UserForm1.Show: Exibe o formulário UserForm1.
- Erro:: Rótulo de erro.
- MsgBox "Erro ao abrir o formulário!", vbCritical, "Erro no Cadastro": Exibe uma mensagem de erro com a descrição do erro ocorrido.

(c) Módulo 2

```
Sub Botão1_Clique()
    Projeto.Show
End Sub
```

Explicando o APÊNDICE A – (c) Módulo 2:

- Sub Botão1_Clique(): Define uma sub-rotina chamada Botão1_Clique. Essa sub-rotina será automaticamente associada ao evento de clique do botão Botão1.
- Projeto.Show: Exibe o formulário chamado Projeto.

(d) Módulo 3

```
Sub Buscar_BD()
    Dim sql As String
    Dim rs As ADODB.Recordset
    Dim i As Long

    ' Conectar ao banco de dados
    Módulo1.ConectarPlan

    ' Consulta SQL
    sql = "SELECT * FROM [Planilha1$]"
```

```
    ' Adicione as cláusulas WHERE necessárias ao SQL conforme necessário

    ' Abrir o recordset com os resultados da consulta
    Set rs = New ADODB.Recordset
    rs.Open sql, Conexao2, adOpenKeyset, adLockReadOnly

    ' Limpar conteúdo da planilha
    Planilha1.Range("A2:C1000000").ClearContents

    ' Copiar dados para a planilha
    Planilha1.Range("A2").CopyFromRecordset rs

    ' Fechar o recordset e liberar memória
    rs.Close
    Set rs = Nothing

    ' Desconectar do banco de dados
    Módulo1.DesconectarPlan
     Sheets("HOME").Select

    ' Mensagem de sucesso
    MsgBox "Busca com sucesso!", vbInformation, "SALVAR"
    Exit Sub

Erro:
    ' Mensagem de erro
    MsgBox "Erro", vbCritical, "SALVAR"
End Sub
```

Explicando o APÊNDICE A – (d) Módulo 3:

- Define uma sub-rotina chamada Buscar_BD. É o início do procedimento onde a busca no banco de dados será executada.

- sql: Declara uma variável do tipo String para armazenar a consulta SQL.
- rs: Declara uma variável do tipo ADODB.Recordset para manipular o conjunto de registros retornados pela consulta.
- i: Declara uma variável do tipo Long, que não está sendo utilizada no código mostrado.
- Chama a sub-rotina ConectarPlan no módulo Módulo1, que estabelece a conexão com as planilhas do Excel que serão usadas como banco de dados.
- Define a consulta SQL que seleciona todos os registros da planilha Planilha1. Um comentário sugere que cláusulas WHERE podem ser adicionadas conforme necessário para filtrar os dados.
- Inicializa um novo ADODB.Recordset.
- Abre o Recordset executando a consulta SQL usando a conexão Conexao2, com as opções adOpenKeyset (conjunto de registros dinâmico) e adLockReadOnly (somente leitura).
- Limpa o conteúdo das células no intervalo de A2 a C1000000 na Planilha1.
- Copia os dados do Recordset rs para a planilha Planilha1, começando na célula A2.
- Fecha o Recordset e libera a memória associada a ele, definindo rs como Nothing.
- Chama a sub-rotina DesconectarPlan no módulo Módulo1, que fecha as conexões com os bancos de dados.
- Seleciona a planilha chamada "HOME".
- Chama a sub-rotina DesconectarPlan no módulo Módulo1, que fecha as conexões com os bancos de dados.
- Seleciona a planilha chamada "HOME".
- Exibe uma mensagem de sucesso indicando que a busca foi realizada com êxito.
- Usa Exit Sub para sair da sub-rotina.
- Rótulo Erro: para o tratamento de erros.
- Exibe uma mensagem de erro caso algum problema ocorra durante a execução da sub-rotina.

APÊNDICE B – Script em Python

```
import serial
import sys
import time

arduino = serial.Serial('COM3', 9600)
time.sleep(2)  # Aguarda o Arduino inicializar

if len(sys.argv) > 1:
    valor = sys.argv[1]
    arduino.write(f"{valor}\n".encode())  # Envia o valor seguido de uma nova linha

arduino.close()
```

APÊNDICE C – Código sketch do Arduino

```
int ledVermelho = 9;
int ledAmarelo = 10;
int ledVerde = 11;
int valorAtual = -1;  // Variável para armazenar o último valor recebido
unsigned long tempoUltimoDado = 0;  // Variável para armazenar o tempo do
último dado recebido
const unsigned long intervalo = 10000;  // Intervalo de tempo em
milissegundos (10 segundos)

void setup() {
  Serial.begin(9600); // Inicia a comunicação serial a 9600 bps
  pinMode(ledVermelho, OUTPUT);
  pinMode(ledAmarelo, OUTPUT);
  pinMode(ledVerde, OUTPUT);

  // Apaga todos os LEDs ao iniciar
  digitalWrite(ledVermelho, LOW);
  digitalWrite(ledAmarelo, LOW);
  digitalWrite(ledVerde, LOW);
}

void loop() {
  if (Serial.available() > 0) {
    int valor = Serial.parseInt(); // Lê o valor inteiro enviado pelo Excel

    // Verifica se o valor é válido e está no intervalo de 0 a 100
    if (valor >= 0 && valor <= 100 && valor != valorAtual) {
      valorAtual = valor;  // Atualiza o valor armazenado
      tempoUltimoDado = millis(); // Atualiza o tempo do último dado
recebido

      // Apaga todos os LEDs antes de acender o correto
      digitalWrite(ledVermelho, LOW);
      digitalWrite(ledAmarelo, LOW);
      digitalWrite(ledVerde, LOW);

      // Condições para acender os LEDs com base no valor
      if (valorAtual <= 40) {
        digitalWrite(ledVermelho, HIGH);
        Serial.println("LED Vermelho aceso");
      } else if (valorAtual > 40 && valorAtual <= 75) {
        digitalWrite(ledAmarelo, HIGH);
        Serial.println("LED Amarelo aceso");
      } else if (valorAtual > 75 && valorAtual <= 100) {
```

```
        digitalWrite(ledVerde, HIGH);
        Serial.println("LED Verde aceso");
      }
    }
  }

  // Desliga os LEDs após 10 segundos se não houver novos dados
  if (millis() - tempoUltimoDado >= intervalo && valorAtual != -1) {
    digitalWrite(ledVermelho, LOW);
    digitalWrite(ledAmarelo, LOW);
    digitalWrite(ledVerde, LOW);
    valorAtual = -1; // Reseta o valor atual
    Serial.println("LEDs desligados após 10 segundos");
  }
}
```