UNIVERSIDADE DO ESTADO DA BAHIA

DEPARTAMENTO DE CIÊNCIAS EXATAS E DA TERRA

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

TAMIRES FARIAS CRUZ

A UTILIZAÇÃO DO ARDUINO COMO FERRAMENTA DE BAIXO CUSTO PARA EXPERIMENTOS DIDÁTICOS DE FÍSICA: DISSIPAÇÃO DE CALOR.

SALVADOR

2021

TAMIRES FARIAS CRUZ

A UTILIZAÇÃO DO ARDUINO COMO FERRAMENTA DE BAIXO CUSTO PARA EXPERIMENTOS DIDÁTICOS DE FÍSICA: DISSIPAÇÃO DE CALOR.

Monografia apresentada ao curso de Sistemas de Informação do Departamento de Ciências Exatas e da Terra da Universidade do Estado da Bahia - UNEB, como requisito para a obtenção do grau de bacharel em Sistemas de Informação.
Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Cláudio Alves de Amorim

SALVADOR

2021

FICHA CATALOGRÁFICA

Sistema de Bibliotecas da UNEB
Dados fornecidos pelo autor

C957a Cruz, Tamires Farias

A UTILIZAÇÃO DO ARDUINO COMO FERRAMENTA DE BAIXO CUSTO PARA EXPERIMENTOS DIDÁTICOS DE FÍSICA: DISSIPAÇÃO DE CALOR. / Tamires Farias Cruz.-- Salvador, 2021.

82 fls : il.

Orientador(a): Prof. Dr. Cláudio Alves de Amorim.

Inclui Referências

TCC (Graduação - Sistemas de Informação) - Universidade do Estado da Bahia. Departamento de Ciências Exatas e da Terra. Campus I. 2021.

1.Arduino. 2. Experimentos de Física. 3. Automação de experimentos.

CDD: 005

TAMIRES FARIAS CRUZ

A UTILIZAÇÃO DO ARDUINO COMO FERRAMENTA DE BAIXO CUSTO PARA EXPERIMENTOS DIDÁTICOS DE FÍSICA: DISSIPAÇÃO DE CALOR.

Monografia apresentada ao curso de Sistemas de Informação do Departamento de Ciências Exatas e da Terra da Universidade do Estado da Bahia - UNEB, como requisito para a obtenção do grau de bacharel em Sistemas de Informação. Área de Concentração: Ciência da Computação

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Cláudio Alves de Amorim (Orientador)
Universidade do Estado da Bahia – UNEB

Prof. Dr. Diego Gervasio Frías Suárez
Universidade do Estado da Bahia – UNEB

Prof. Dr. Leandro Santos Coelho de Souza
Universidade do Estado da Bahia – UNEB

Dedico este trabalho à minha amada mãe Terezinha, mulher corajosa e guerreira, minha inspiração e meu maior orgulho. Dedico também ao meu pai Leandro (in memoriam), meu eterno herói de quem eu sinto muita saudade, sei que de seu lugar cuida de mim.

AGRADECIMENTOS

Sou grata a Deus por em meio a uma pandemia ter preservado a minha saúde e a de toda a minha família, permitindo com que eu pudesse terminar este trabalho com um pouco mais de tranquilidade e por ter me guiado em toda essa jornada me dando força e determinação para sempre seguir em frente e nunca desistir.

Agradeço a minha mãe Terezinha, por sempre priorizar uma educação de qualidade para mim, por aguentar todo o meu estresse e drama, por ser uma amiga fiel e estar sempre ao meu lado para o que der e vier. Eu não tenho palavras para expressar o tamanho da minha gratidão por sempre confiar e acreditar na minha capacidade quando nem eu mesma acreditava. Agradeço ao meu pai Leandro (in memoriam), por me causar lembranças felizes todas as vezes em que eu trabalhei com a parte prática deste projeto. Sempre muito inteligente e curioso com certeza ele ficaria empolgadíssimo com o Arduino e suas possibilidades.

Não posso deixar de lembrar e agradecer a minha tia Maria Izabel e a minha tia e madrinha Noélia, por serem tão presentes na minha vida, por me encherem de amor e carinho e sempre torcerem para o meu sucesso e crescimento.

Agradeço ao meu orientador neste projeto, Prof. Dr. Cláudio Alves de Amorim, por toda a dedicação, comprometimento, ajuda, disposição e paciência. Mesmo com todos os problemas e dificuldades que esse ano de 2020 trouxe a todos nós, em nenhum momento deixou de apoiar e incentivar da melhor forma possível. Não foi surpresa para mim a excelência da sua orientação, eu tinha certeza que seria assim desde o início, muito obrigada por esse privilégio.

Agradeço também a Profª. Msc. Débora Alcina Rego Chaves, por ser tão prestativa e gentil, obrigada por suas orientações de madrugada, por todo o seu esforço, esmero e perfeccionismo.

Agradeço aos meus amigos e colegas da UNEB, por enfrentarem os desafios da universidade junto comigo e tornarem as situações difíceis mais leves, por serem sérios e responsáveis quando preciso e divertidos e doidos quando não preciso. Enfim, agradeço a todos que de alguma forma participaram dessa jornada e me ajudaram a conquistar esse objetivo.

RESUMO

Este trabalho apresenta um estudo realizado com o objetivo de desenvolver e validar um ambiente aberto (software e hardware) para experimento didático de Física, utilizando como instrumentação de prototipagem eletrônica a plataforma Arduino. Como prova de conceito para esse ambiente aberto e automatizado, foi proposto um experimento didático na área de Termodinâmica, especificamente de dissipação de calor. O uso do aparato experimental proposto, buscou intermediar o processo de obtenção de conhecimento, aliando a teoria e a prática e o processamento computacional de aquisição das grandezas físicas medidas. A intenção foi que esse ambiente fosse capaz de minimizar a preocupação com o monitoramento do experimento e registro dos dados, automatizando essas tarefas. A fim de validar a utilidade do ambiente, foram realizados experimentos para analisar o calor dissipado pelo componente eletrônico transistor. O desenvolvimento do ambiente seguiu as etapas de: construção do circuito eletrônico, implementação de sistema de monitoramento executado no Arduino e, por fim, implementação de sistema de gerenciamento, codificado na linguagem Python. Após a análise dos resultados obtidos, constatou-se que o objetivo do trabalho foi alcançado, uma vez que o ambiente de hardware e software apresentou eficiência do ponto de vista didático, fornecendo dados concisos sobre o experimento em questão e incentivando um olhar investigativo e crítico sobre o fenômeno físico estudado.

Palavras-chave: Arduino. Experimentos de Física. Automação de experimentos. Aprendizagem Significativa.

ABSTRACT

This work presents a study carried out with the objective of developing and validating an open environment (software and hardware) for didactic experiment in Physics, using the Arduino platform as an instrumentation for electronic prototyping. As a proof of concept for this open and automated environment, a didactic experiment was proposed in the area of thermodynamics, specifically heat dissipation. The use of the proposed experimental apparatus, sought to mediate the process of obtaining knowledge, combining theory and practice and computational processing for the acquisition of the measured physical quantities.The intention was that this environment could minimize the concern to monitor the experiment and record the data, automating these tasks.To validate the usefulness of the environment, experiments were carried out to analyze the heat dissipated by the electronic component transistor.The development of the environment followed the steps of: construction of the electronic circuit, implementation of a monitoring system implemented in Arduino and, finally, implementation of a management system, coded in Python. After analyzing the results obtained, it was found that the objective of the work was achieved, since the hardware and software environment showed efficiency from the didactic point of view, providing concise data about the experiment in question and stimulating an investigative and critical look about the study of the physical phenomenon.

Keywords: Arduino. Physics experiments. Automation of experiments. Meaningful Learning.

LISTA DE FIGURAS

Figura 1 – Maior agitação entre as moléculas, maior temperatura. 22
Figura 2 – Representação para resistor com uma resistência R. 23
Figura 3 – Modos de propagação de calor com a utilização de dissipador. 24
Figura 4 – Junções NPN e PNP em um Transistor, e seus respectivos símbolos. 26
Figura 5 – Conexão de transistor em Emissor-Comum(EC) 28
Figura 6 – IDE do Arduino, e funções básicas 30
Figura 7 – Descrição dos terminais e principais componentes da placa Arduino UNO R3. 32
Figura 8 – Pinos do transistor TIP31 e instalação em dissipador de calor. 35
Figura 9 – Sensor de temperatura DS18B20. 36
Figura 10 – Polarização de transistor em Emissor-Comum(EC) 36
Figura 11 – Esquema final do experimento. 39
Figura 12 – Circuito físico do experimento. 40
Figura 13 – Arquitetura do ambiente de *hardware* e *software*. 41
Figura 14 – Sistema de Gerenciamento - SG. 45
Figura 15 – Selecionar a porta de comunicação serial - IDE ARDUINO. 47
Figura 16 – SG - Alertas para temperatura máxima e interrupção do experimento. 49
Figura 17 – Arquivo .csv gerado pelo SG. 49
Figura 18 – Sistema de Gerenciamento - Exibição Gráfico de Temperaturas 52
Figura 19 – Gráfico de temperaturas para os Experimentos 1 e 2. 54
Figura 20 – Gráfico de temperaturas para os Experimentos 1 a 6. 55
Figura 21 – Fotos - Ambiente *Hardware* e *Software* 65

LISTA DE TABELAS

Tabela 0 – Valores elétricos máximos do TIP31C. 37
Tabela 1 – Variáveis de entrada. 53
Tabela 2 – Conjunto de resultados. 53

LISTA DE CÓDIGOS-FONTE

Código-fonte 1 – SM - Inclusão de bibliotecas e definição dos pinos 41
Código-fonte 2 – SM - Função de configuração *void setup()* 42
Código-fonte 3 – SM - Função *void loop()* 43
Código-fonte 4 – SG - Utilização da biblioteca *pySerial* 46
Código-fonte 5 – SG - Loop de leitura serial das temperaturas 47
Código-fonte 6 – SG - Loop para plotagem do gráfico de temperaturas 50

LISTA DE SÍMBOLOS

P_D	Potência dissipada
$P_D(mx)$	Potência máxima dissipada
V_{EC}	Tensão Emissor-base
I_C	Corrente do Coletor
R	Resistência
V	Volt
A	Ampère
W	Watt

SUMÁRIO

1 INTRODUÇÃO 14
2 REVISÃO DE LITERATURA 18
3 GERAÇÃO DE CALOR EM COMPONENTES ELETRÔNICOS . . . 21
3.1 Temperatura e Calor 21
3.1.1 Condução 22
3.1.2 Convecção 22
3.1.3 Irradiação 23
3.2 Dissipação de Calor 23
3.2.1 Lei de Joule 24
3.2.2 Dissipação de Calor no Transistor 26
3.2.2.1 Estrutura e Funcionamento 26
3.2.2.2 Cálculo de dissipação 27
4 DISPOSITIVO MICROCONTROLADO ARDUINO 29
4.1 Definição 29
4.2 Plataforma de desenvolvimento 30
4.2.1 *Software* 30
4.2.2 *Hardware* 31
5 METODOLOGIA 34
5.1 Ambiente aberto para experimento didático de dissipação de calor . . . 34
5.1.1 Planejamento do Experimento 34
5.1.1.1 Circuito de polarização do TIP31C 36
5.1.1.2 Parâmetros Experimentais 38
5.1.2 Implementação do *software* 40
5.1.2.1 Desenvolvimento do Sistema de Monitoramento 41
5.1.2.2 Desenvolvimento do Sistema de Gerenciamento 44
5.1.3 Aquisição e Análise dos Dados 52
6 CONCLUSÃO E TRABALHOS FUTUROS 58
REFERÊNCIAS 60

APÊNDICES 64
APÊNDICE A – AMBIENTE *HARDWARE* E *SOFTWARE* **65**
APÊNDICE B – CÓDIGO - SISTEMA DE MONITORAMENTO SM . . **66**
APÊNDICE C – CÓDIGO - SISTEMA DE GERENCIAMENTO SG . . . **69**

1 INTRODUÇÃO

Segundo Osvaldo Frota-Pessoa:

> (...) só quando o estudante está pesquisando fatos reais, que efetivamente se estão desenrolando perante ele (e não imaginariamente no quadro negro), só quando investiga, aguçado pela curiosidade e pelo encantamento ante o mistério, está ele aprendendo ciência. (1952, apud MOREIRA; PAIVA, 2016, p. 63).

A forma como a Física é ministrada nas escolas e universidades do Brasil, tradicionalmente é alvo de vários estudos a respeito da estrutura, dos métodos utilizados e do processo de ensino-aprendizagem (MOREIRA, 2018)(ORGANTINI, 2018)(PETRY et al., 2016)(SARIK; KYMISSIS, 2010)(SILVEIRA; GIRARDI, 2017). Isso se deve não só à tradição em pesquisas na área, como também à busca por alternativas para melhorar o aprendizado dessa disciplina, que historicamente, é considerada por boa parte dos alunos como desinteressante, enfadonha e de difícil compreensão.

Há uma tendência das pesquisas em descobrir artifícios para modificar as práticas didáticas aplicadas no ensino de Física, que no geral, não fomentam a aprendizagem significativa, como também, não instigam o cunho investigativo e crítico necessário para o ensino de uma ciência. Moreira (2018), ao citar alguns aspectos que descrevem o ensino da Física tanto no ensino médio como no superior, confirma os problemas na didática aplicada, quando diz que um dos motivos favoráveis a forte indisposição dos alunos para aprender sobre Física é o fato da disciplina ter características metodológicas desmotivantes, centradas no docente, na memorização de fórmulas, na resolução de problemas conhecidos e na preparação para testagem. Além disso, dentre os maiores obstáculos na aprendizagem, está a complexidade em relacionar ao cotidiano dos alunos, os conceitos físicos aprendidos no ambiente educacional. Segundo Heckler et al. (2007, p. 267) “Para a maior parte dos alunos, a física não passa de um conjunto de códigos e fórmulas matemáticas a serem memorizadas e de estudos de situação que, na maioria das vezes, estão totalmente alheias às suas experiências cotidianas”.

Sendo assim, se torna muito custoso estabelecer um vínculo entre a teoria e prática, no cenário em que as situações-problema apresentadas em sala de aula e descritas nos livros, são mecânicas e dificilmente se encaixam e fazem sentido para a realidade dos alunos. Essa situação

colabora para um total desânimo e falta de percepção para a importância do estudo da disciplina, como também, impossibilita os alunos de participarem ativamente do processo de construção do conhecimento investigativo e crítico (GRASSELLI; GARDELLINI, 2016).

Nesse aspecto, considerando que na sociedade atual os alunos tem contato com equipamentos tecnológicos desde a infância (LARA et al., 2013), a desconexão entre o cotidiano do aluno e a Física instruída em sala de aula, expõe dentre outras causas, a ineficiência na inserção de Tecnologias de Informação e Comunicação, assim como, um equívoco na aplicação de experimentações práticas, no ambiente educacional (MOREIRA, 2018)(ROSA; TRENTIN; BIAZUS, 2017). Todavia apesar da carência na utilização de tecnologias contemporâneas no ensino de Física, atualmente existe o intuito em mudar essa realidade. Estudos são realizados acerca dessa temática, sobretudo com a intenção de incorporar cada vez mais a tecnologia em dinâmicas práticas, a fim de tornar o ambiente escolar e universitário mais atrativo para o discente. Esses trabalhos evidenciam a utilização do microcomputador como ferramenta cognitiva, essencial para os processos de ensino-aprendizagem, especialmente na automação de experimentos didáticos (BEZERRA et al., 2009).

Contudo ainda existem barreiras sobre a real utilização de tecnologia, isso se deve principalmente a pouca oferta no mercado nacional e ao custo elevado dos kits de laboratórios oferecidos pelos principais fornecedores comerciais da área de experimentos. Como alternativa, verificamos uma tendência das pesquisas relacionadas à automação de experimentos via microcomputador, em propor soluções de baixo custo envolvendo diferentes portas de comunicação e periféricos (SOUZA et al., 2011). Nessa linha, dentre as soluções de baixo custo mais utilizadas, o Arduino surge como a mais popular e acessível (MARTINAZZO et al., 2014)(MOREIRA et al., 2018). Apresentando uma tecnologia versátil e de simples utilização por professores e alunos, esse microcontrolador mostra-se como boa alternativa, por ser uma plataforma eletrônica de código aberto fundamentada em hardware software fáceis de usar, e com o custo relativamente baixo (CAVALCANTE; TAVOLARO; MOLISANI, 2011).

Posto isto, ao verificar a produção científica acerca da utilização do Arduino em laboratórios e/ou experimentos didáticos de física, pudemos averiguar a sua aplicabilidade, potencialidade, benefícios e limitações quando empregado nesse contexto. Observamos que a maior parte dos trabalhos cita como benefícios o fato do Arduino ser uma ferramenta de baixo-custo, versátil e de fácil utilização. Essas características fazem com que essa placa eletrônica, e

de uso genérico, permita que o seu controle e aquisição de dados, sejam ideais para serem usados como parte de dispositivos de laboratório de física e engenharia (BUKSMAN et al., 2019). Além disso, a versatilidade com que o Arduino envolve diferentes portas de comunicação e periféricos do PC, possibilita (através da utilização de vários sensores) que ele seja um microcontrolador que potencializa suas funções para além de uma simples interface passiva (SOUZA et al., 2011).

Quanto a temática dos experimentos didáticos automatizados, verificamos a presença de todas áreas tradicionais da Física, com atenção especial para um expressivo aumento, ao longo dos anos, na quantidade de experimentos na área da Termodinâmica. Enquanto no trabalho de revisão sistemática realizado por Araújo e Veit (2004), os experimentos relacionados a Termodinâmica aparecem em menor número (14% de um total de 101 artigos) na revisão feita por Moreira et al. (2018), experimentos em Termodinâmica são maioria, representando 40% dos achados. Nesse sentido, buscamos trabalhos que relacionassem o Arduino com os experimentos nessa área, especificamente experimentos em dissipação de calor, já que os componentes eletrônicos utilizados com esse microcontrolador têm relação direta com tal fenômeno. Entretanto, houve dificuldade em encontrar trabalhos que lidassem com esse tipo específico de experimento. Apenas o trabalho de Buksman et al. (2019), apresentou um estudo próximo ao tema.

Perante o exposto, a partir do momento em que se tem contato com tecnologia atual e as preocupações com coletas de dados e monitoramento são minimizadas, a utilização do Arduino abre várias possibilidades para uma maior análise crítica e investigativa do experimento em si. Dessa maneira, diagnosticamos a existência de um espaço de pesquisa para a construção de um ambiente de baixo custo e mais significativo para a aprendizagem do aluno. Somando-se a isso, a pouca presença de experimentos didáticos que tratam o fenômeno da dissipação de calor o problema dessa pesquisa apresenta-se da seguinte forma: A escassez de experimentos automatizados de Física, com a utilização do microcontrolador Arduino, que tenham como foco uma aprendizagem significativa do tema: dissipação de calor.

Deste modo o objetivo do presente trabalho visa o desenvolvimento e a validação de um ambiente significativo aberto (*hardware* e *software*), baseado no microcontrolador Arduino, para experimentos didáticos de Física, usando como prova de conceito um experimento de dissipação de calor. Para alcançar esse objetivo foram definidos os seguintes objetivos específicos:

- Desenvolver o desenho do experimento, construir e montar o aparato experimental;

- Definir os parâmetros experimentais;
- Configurar e implementar as funcionalidades do software de monitoramento do aparato experimental no Arduino;
- Implementar software de gerenciamento do experimento;
- Realizar os testes de validação do ambiente;
- Analisar os resultados obtidos;

Essa pesquisa pretende acrescentar no âmbito científico, no que se diz respeito à utilização de um ambiente para experimentos didáticos significativo, uma possibilidade de aproximação entre a didática da ciência aplicada em sala de aula, com as pesquisas científicas. Nesse quadro, é importante dentre outros fatores, aproximar o aluno da construção do conhecimento, estimulando-o com utilização de tecnologias atuais, com o intuito que ele participe do processo de formação da teoria, das práticas experimentais e dos problemas. A participação ativa do aluno na concepção e no entendimento do aparato experimental, especialmente no ensino de Física, colabora com a aprendizagem significativa, a partir do ponto em que há um distanciamento dos métodos de ensino ultrapassados e inflexíveis. Ao trabalhar com aparatos experimentais, tecnológicos, automatizados e igualmente acessíveis aos professores e alunos, criamos um cenário em que existe a possibilidade deles, em parceria, construírem e modificarem todo um ambiente de ensino-aprendizagem, onde a dúvida, os questionamentos e as conclusões motivam uma visão mais representativa.

No âmbito sócio-econômico a utilização do Arduino no ensino de Física, um equipamento acessível, de baixo-custo e de fácil instrução, proporciona aos estudantes de vários níveis acadêmicos e sociais o acesso ao aprendizado crítico e investigativo de uma ciência da natureza. A visão mais profunda sobre os fenômenos e tecnologias que os cercam, faz com que desenvolvam um melhor entendimento da sua realidade, do seu cotidiano e do seu mundo. Por fim, no âmbito acadêmico, a presente pesquisa justifica-se no sentido de colaborar com a disseminação de novos estilos de ensino-aprendizagem para o ensino da Física, e mais especificamente, por contribuir na viabilização de um artefato experimental micro controlado que lida com o estudo de uma experiência teórico-prática pouco automatizada: a dissipação de calor.

2 REVISÃO DE LITERATURA

No geral, o estado da arte revelou um consenso sobre a utilização do Arduino como instrumento facilitador da aquisição automática de dados. A maioria dos trabalhos apontam experimentos, automatizados e associados a sistemas ou aplicativos, direcionados a exibição dos dados coletados (SOUZA et al., 2011)(LESTEIRO-TEJEDA; HERNÁNDEZ-DELFÍN; BATISTA-LEYVA, 2017)(LLAMAS et al., 2018)(HILBERER et al., 2018). O fato do Arduino ser uma alternativa de baixo custo, também é mostrado como um grande benefício, assim como, sua associação com ferramentas como Labview, Scilab, Matlab e com outros microcontroladores, principalmente o Raspberry Pi (FERNÁNDEZ-PACHECO; MARTIN; CASTRO, 2019)(ASHA et al., 2017)(ARADI, 2016). Isso salientou as diversas possibilidades de incremento e modificação do Arduino tanto a nível de *hardware* quanto a nível de *software* (SOUZA et al., 2011). No contexto da utilização em Física por exemplo, o funcionamento do Arduino, baseia-se na leitura simultânea de sinais elétricos de dezenas de sensores tanto digitais quanto analógicos. Isso é de grande serventia, pois, com esses sensores é possível ler e mensurar dados de vários fenômenos físicos detectáveis (MARTINAZZO et al., 2014).

Em relação a limitações deste microcontrolador, não houve muitas observações. Entretanto, a preocupação de alguns autores com a precisão dos sensores do Arduino, chamou a atenção. Koestoer et al. (2019) propõem a calibração do sensor de temperatura DS18B20. Mencionam que o Arduino pode ser um dispositivo de aquisição de dados válido apenas se o sensor estiver perfeitamente calibrado. Ao final da aplicação do método proposto, foi demonstrado que a acurácia do sensor melhorou em cerca de 0,85%, em relação a medições anteriores, tornando assim o sensor mais confiável e preciso. Similarmente, Vilar et al. (2015), apresentam medidas e comparações do tempo morto (tempo mínimo necessário para que o sensor comece a indicar variações da propriedade física estudada) do sensor de temperatura DS18B20 e do termistor NTC. Ao final do estudo, os resultados mostraram que o tempo morto dos sensores estudados devem ser um fator não desprezível, já que os dados obtidos podem não refletir a realidade.

Quanto aos trabalhos filtrados para atender o critério de experimentos voltados a termodinâmica (dissipação de calor), observamos que esses apresentaram em maioria, expe-

rimentos de propagação de calor por condução térmica. Amorim et al. (2015) mostram uma experiência clássica de condução térmica usando sensores de temperatura digitais, associados a uma placa Arduino para controle e aquisição de dados. O experimento consistiu na condução de calor através de uma barra metálica cilíndrica conectada em suas extremidades com duas fontes térmicas mantidas a temperaturas constantes. O objetivo foi comparar variáveis reais e teóricas sobre a lei de Fourier. O Arduino foi utilizado para automação da aquisição dos dados, de forma que facilitasse a realização do experimento e a análise das variáveis em intervalos determinados de tempo.

Ainda analisando os experimentos de termodinâmica, existiu uma dificuldade em encontrar artigos que focassem no fenômeno dissipação de calor. Apenas um artigo em um total de 155 analisados durante a revisão bibliográfica, apresentou um estudo próximo ao tema. Buskman et al. (2019), mostraram a montagem de um dispositivo, para experimentos na área de propagação de calor através de uma barra metálica. Nesse caso, o Arduino é utilizado junto ao ambiente gráfico Xcos da Scilab, e não participa apenas como instrumento de automação. O aparato experimental é planejado de forma que os próprios componentes eletrônicos do microcontrolador tenham participação ativa na experiência. Sendo assim, a dissipação de calor de um transistor TIP41C é usada como fonte de calor para a barra metálica do experimento, provando com isso, a grande versatilidade e as possibilidades de uso do Arduino.

Diante dos achados da revisão de literatura realizada, a maioria dos experimentos abordados, são associados a sistemas ou aplicativos simples, normalmente para a ocasião do experimento, e direcionados a exibição dos dados coletados. Percebemos que há pouca incidência de estudos para a construção de sistemas mais genéricos que atendam a mais de um experimento. Em outro ponto, verificamos que vários trabalhos utilizam o Arduino como forma de provar teorias consolidadas nos livros didáticos. Nos artigos lidos, normalmente o resultado para esse tipo de verificação era divergente, já que a sensibilidade dos sensores utilizados, tinha que ser levada em consideração para a comparação teórico-prática. Sobre a abordagem dos experimentos físicos de termodinâmica, a tendência é replicar, usando o Arduino, experimentos clássicos de sala de aula. O assunto principal abordado foi a propagação de calor via condução térmica, e o sensor de temperatura utilizado com maior frequência foi o DS18B20.

Por fim a revisão foi rica no sentido de difundir as tecnologias associadas ao Arduino, assim como a utilização do microcontrolador em projetos educacionais. O acesso a livros

didáticos sobre o Arduino também foi de grande importância para a realização do presente trabalho. Os livros Arduino Básico (MCROBERTS, 2015) e Arduino Cookbook (MARGOLIS, 2011) destacam-se por direcionar, a utilização de variados tipos de sensores e atuadores com aplicações gerais em projetos exemplos.

No capítulo seguinte serão analisados conceitos da área de física importantes para a concepção e construção do ambiente de *hardware* e *software* apresentado nesse trabalho. Especificamente serão discutidas as relações entre temperatura e calor, meios de condução térmica e conceitos relacionados a dissipação de calor no contexto da eletrônica.

3 GERAÇÃO DE CALOR EM COMPONENTES ELETRÔNICOS

No campo da Eletrônica, parâmetros relacionados a temperatura e calor, têm expressiva importância sobre o correto e seguro funcionamento dos projetos (BRAGA, 2014). O ambiente para experimentos automatizados de Física, implementado no presente trabalho, tem como parte fundamental da sua arquitetura, circuito eletrônico construído com base no funcionamento de um dos componentes mais importantes da área: O transistor. O estudo da dissipação de calor proposto no experimento, tem como referência, o comportamento térmico desse transistor, sob diferentes condições experimentais relacionadas as grandezas elétricas aplicadas no circuito. Sendo assim, esse capítulo, tem por objetivo apresentar conceitos básicos sobre a geração de calor, no contexto dos circuitos e componentes eletrônicos.

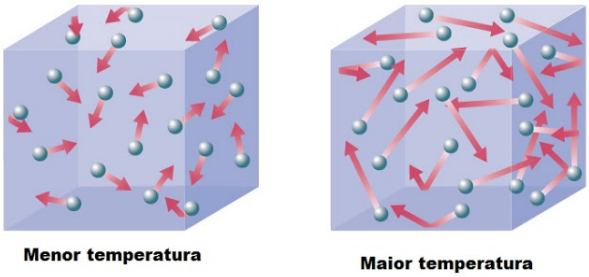
3.1 TEMPERATURA E CALOR

Segundo Young e Freedman (2016) as palavras "temperatura"e "calor"socialmente costumam ser utilizadas como sinônimos na linguagem cotidiana. Entretanto, em Física, esses dois termos têm significados bastante diferentes. Na Termologia, parte da Física em que estudamos os fenômenos ligados a energia térmica, esses fenômenos, assim como a maioria dos eventos da natureza, podem ser interpretados tanto sobre o ponto de vista macroscópico, quanto pelo ponto de vista microscópico. Para o contexto aplicado à Eletrônica, é interessante saber que a visão microscópica sobre os conceitos de temperatura e calor, é sem dúvida a mais importante. Com isso, a partir do entendimento que toda matéria é constituída por moléculas em constante movimento, e que a energia cinética associada a esse movimento é denominada energia térmica, podemos concluir que temperatura nada mais é do que o grau de agitação das moléculas de um corpo. Quando então, constatamos que um corpo está "mais quente"do que outro é porque seus átomos se agitam com mais intensidade, conforme mostra a Figura 1.

A agitação das moléculas se deve ao fato de que elas possuem energia cinética. Assim, seguindo o mesmo raciocínio, a quantidade de energia armazenada pelas moléculas da matéria nos permite correlacionar a outra grandeza em questão: o calor. Portanto, calor é definido como a energia que devemos entregar a um corpo, para que seus átomos/moléculas

se movimentem e sua temperatura se modifique. Na prática, o calor pode ser transferido a um corpo por três mecanismos: condução, convecção e radiação (BRAGA, 2012b)(RAMALHO; NICOLAU; TOLEDO, 2015)(YOUNG; FREEDMAN, 2016).

Figura 1 – Maior agitação entre as moléculas, maior temperatura.



Fonte – Retirada da página Saber Atualizado (2020).

3.1.1 Condução

Nesse mecanismo, a energia térmica presente no corpo, propaga-se em virtude da agitação molecular no interior de um material, nesse caso é necessário contato entre os corpos, pois, o calor é transferido através da matéria, do sentido de maior agitação das moléculas (mais quente), para o de menor agitação (mais frio). Esse processo de condução, ocorre devido as colisões entre átomos vizinhos, que fazem com que a parte da energia cinética seja transferida de um átomo a outro. Os átomos, em si, não se deslocam de uma região a outra do material, e sim a energia envolvida entre seus choques (RAMALHO; NICOLAU; TOLEDO, 2015)(YOUNG; FREEDMAN, 2016).

3.1.2 Convecção

Na convecção, a transferência de calor ocorre devido ao movimento da massa de um fluido (líquidos ou gases) de uma região no espaço para outra. Esse movimento geralmente é ocasionado pela diferença entre as densidades dos fluidos envolvidos. Por exemplo, o contato de um corpo aquecido com o ar faz com que esse corpo transfira energia através da troca de gases. Isso acontece, por consequência do ar mais próximo ao corpo, com partículas mais agitadas e consequentemente menos densas, formar uma corrente de ar que transfere o calor gerado (RAMALHO; NICOLAU; TOLEDO, 2015)(YOUNG; FREEDMAN, 2016).

3.1.3 Irradiação

Na radiação a transferência de calor é feita por meio de ondas eletromagnéticas, como a luz visível, a radiação infravermelha e a radiação ultravioleta. Também conhecida como irradiação, esse mecanismo é uma forma de transferência de calor que ocorre por meio de ondas eletromagnéticas que podem propagar-se no vácuo, não sendo necessário o contato entre os corpos para haver a transferência de calor (RAMALHO; NICOLAU; TOLEDO, 2015)(YOUNG; FREEDMAN, 2016).

3.2 DISSIPAÇÃO DE CALOR

Aproximando os conceitos citados anteriormente ao contexto de Eletrônica, segundo Newton C. Braga:

> Todo dispositivo eletrônico, que não apresente uma resistência nula, gera uma certa quantidade de calor ao ser percorrido por uma corrente elétrica. Como o dispositivo de resistência nula é ideal, não existindo na prática, podemos dizer que todos os dispositivos percorridos por corrente, num circuito real, geram calor. (2014, p. 325).

Entende-se por resistência como a propensão dos materiais de dificultar o fluxo de corrente ou, mais especificamente, o fluxo de carga elétrica. Dessa forma, os elétrons que compõem a corrente elétrica interagem com a estrutura atômica do material no qual estão se movimentando, a estrutura atômica, por sua vez, resiste a essa interação. No decurso desse processo, uma parte da energia elétrica é convertida em energia térmica e dissipada sob a forma de calor. Nesse sentido, os dispositivos eletrônicos têm a alternativa de aproveitar esse calor gerado para o próprio funcionamento do sistema, ou eliminar a temperatura em excesso, com o intuito de proteger o correto funcionamento dos componentes (BRAGA, 2012b). O componente elétrico de circuito usado para modelar a capacidade de resistência é o resistor. A Figura 2 mostra a simbologia - em padrão americano - para a representação de um resistor no circuito, onde R denota o valor da resistência do resistor.

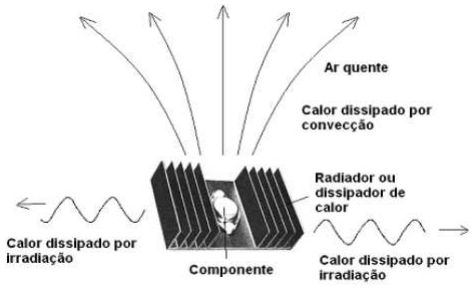
Figura 2 – Representação para resistor com uma resistência R.

Fonte – Autor.

Para os casos em que o calor gerado é inadequado para a segurança funcional do componente eletrônico, é necessária a utilização de meios auxiliares para o descarte dessa energia em excesso. Isso deve ser pensado de forma que o componente não ultrapasse os limites toleráveis de temperatura determinados em sua fabricação.

Um dos principais meios que facilitam a eliminação de calor nos circuitos eletrônicos é a utilização dos dissipadores de calor. Essas estruturas são fixadas aos componentes que geram calor, que por sua vez, através do contato, transferem a energia dissipada por condução térmica à estrutura. Após a transferência do calor entre o componente e o dissipador, o processo de eliminação da energia térmica segue basicamente por duas formas: irradiação e convecção (BRAGA, 2014). O dissipador, responsável por descartar o calor para o meio ambiente, faz com que uma parcela do calor seja irradiada na forma de ondas eletromagnéticas, e a outra parcela seja transferida para o ar através do contato com a sua superfície. Para uma melhor performance, é muito importante que o dissipador tenha uma área de contato para o ar planejada de forma que exista um caminho livre para sua circulação. Na Figura 3 mostramos os dois modos de propagação de calor realizados pelo dissipador, citados anteriormente.

Figura 3 – Modos de propagação de calor com a utilização de dissipador.



Fonte – Retirada de Newton C. Braga (2014).

3.2.1 Lei de Joule

A quantidade de calor que um resistor produz, quando percorrido por uma corrente, é calculada pela Lei de Joule. Essa equação aplica-se a qualquer componente eletrônico e basicamente estabelece que a potência dissipada, ou quantidade de calor gerado é igual a voltagem através do componente, multiplicado pela corrente que passa por aquele componente

(MALVINO; BATES, 2016)(BRAGA, 2012a), conforme a seguinte fórmula:

$$P = V \times I, \tag{3.1}$$

Onde: P é a potência em watts (W), V é a tensão em volts (V) e I é a corrente em amperes (A)

A Lei de Joule é bastante utilizada para a escolha de componentes eletrônicos (BRAGA, 2014), esses componentes geralmente exibem em suas folhas de dados os seus valores nominais máximos de potência. Na escolha de um componente, é importante considerar esses valores, pois, a operação em níveis de potência mais altos do que os nominais, podem danificar ou inutilizar a peça, fazendo com que a mesma opere de forma incorreta ou simplesmente queime. Com isso, é necessário determinar o máximo de corrente que passará pela peça e a voltagem através da mesma, multiplicar esses valores e comparar com a potência nominal máxima fornecida pelo folha de dados em questão. É importante atentarmos também, às condições em que essa potência nominal foi determinada, analisando em conjunto com o seu valor máximo, parâmetros como a temperatura ambiente e temperatura do invólucro do componente em questão.

Por conseguinte, observando a Lei de Ohm onde a corrente em um resistor é proporcional a tensão em seus terminais, ou seja, R = V / I. A Lei de Joule pode ser derivada, de forma a determinar a potência, sem necessariamente conhecer os valores de tensão(V). Com o valor da resistência(Ohm) e da corrente(A) a equação estabelece-se da seguinte maneira (BRAGA, 2012b):

$$P = R \times I^2, \tag{3.2}$$

Segundo Malvino e Bates (2016) a eletrônica atualmente é dividida em vários ramos, um dos mais importantes é sem duvida, aquele que estuda especificamente os sistemas de potência. A Eletrônica de potência como é conhecida, trata dos componentes e circuitos que operam com correntes intensas e consequentemente tensões elevadas que podem chegar a milhares de volts. O Transistor Bipolar de Junção (BJT – *Bipolar Junction Transistor*), nesse contexto, aparece como importante componente de potência para boa parte das aplicações tecnológicas atuais. A seguir,

serão descritas suas características básicas de estrutura, funcionamento, e claro seus parâmetros relacionados a dissipação de potência ou calor.

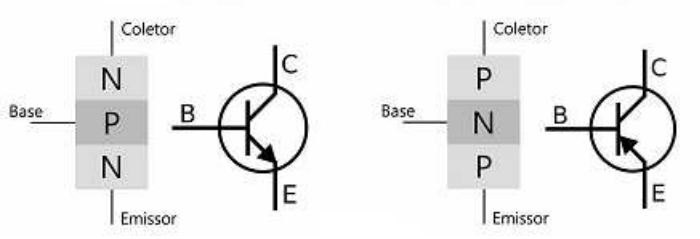
3.2.2 Dissipação de Calor no Transistor

Em 1951, William Schockley inventou o primeiro transistor de junção. Esse componente formado por diferentes regiões semicondutoras, e com a capacidade de amplificar sinais elétricos foi responsável por toda uma revolução na eletrônica, servindo de base para o funcionamento de boa parte dos dispositivos tecnológicos atuais. O que conhecemos hoje como "chip", nada mais é do que uma pastilha de CI (circuito integrado), contendo milhares e até mesmo milhões de transistores. Isto significa que praticamente todo o comportamento funcional dos equipamentos eletrônicos modernos, em visão geral, depende de transistores (MALVINO; BATES, 2016)(SCHULER, 2013).

3.2.2.1 Estrutura e Funcionamento

O Transistor Bipolar de Junção (BJT), é um dispositivo eletrônico semicondutor, geralmente feito de germânio ou silício, usado basicamente para duas funções: amplificar sinais elétricos ou agir como uma chave eletrônica em circuitos. Na sua estrutura fundamental, geralmente é composto de três terminais. São eles: Emissor, Base e Coletor (BRAGA, 2012a)(BRAGA, 2014).

Figura 4 – Junções NPN e PNP em um Transistor, e seus respectivos símbolos.



Fonte – Retirada de página Tecdicas (2020).

Como observado na Figura 4, para obter uma estrutura equivalente a um transistor, devemos agregar três regiões semicondutoras de polaridades alternadas, de modo que entre elas existam duas junções. Nesse caso, a camada N representa o material que possui elétrons livres e a camada P representa o material que possui as lacunas para onde esses elétrons podem ir. Os terminais Emissor e Coletor representam as camadas externas e a base representa a camada do meio, formando assim dois tipos de configurações: NPN e PNP (SCHULER, 2013).

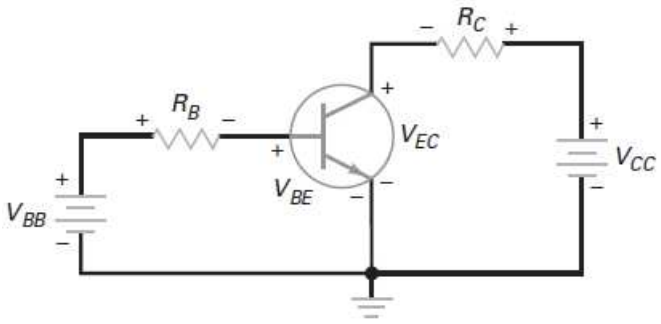
Em relação ao funcionamento, Charles Schuler (2013) explica que os transistores bipolares atuam de três formas diferentes de acordo com as funções que irão desempenhar no circuito. Essas formas são as regiões de operação do transistor denominadas de: corte, ativa ou saturação. Para configurar cada região, é necessário realizar a polarização do transistor, que nada mais é do que aplicar em seus terminais tensões de polaridades apropriadas que o levem às condições normais de funcionamento, ou seja, calcular as tensões e correntes que serão aplicadas ao componente para que ele funcione adequadamente em uma determinada região de operação.

Quando na região ativa, o transistor é utilizado geralmente como amplificador, o circuito é configurado para que a corrente elétrica coletor-emissor seja regulada de acordo com a passagem de corrente pela base. Nas região de corte e saturação, o transistor funciona como chave aberta e fechada respectivamente. Como chave aberta, o transistor impede a corrente de circular através da base fazendo com que também não haja corrente entre coletor e emissor. Já como chave fechada, a corrente que circula pelo terminal da base está no nível máximo, fazendo com que o transistor, entre coletor e emissor, se comporte como um interruptor fechado (SCHULER, 2013)(MALVINO; BATES, 2016).

3.2.2.2 Cálculo de dissipação

De acordo com Newton C. Braga (2014), existem três tipos de configuração utilizados para conectar um transistor: em EC (emissor comum), CC (coletor comum) ou BC (base comum). Por ser uma configuração mais utilizada, e que proporciona maiores ganhos de potência, a configuração EC, foi escolhida para a polarização do transistor utilizado no presente trabalho. Conforme a Figura 5, percebemos que o lado terra de cada fonte de tensão está conectado ao emissor, por isso, o circuito é caracterizado por configuração em emissor comum EC. Esse circuito em específico possui duas malhas, e está configurado de forma que ao variar a corrente de base, através dos valores de V_{BB} e R_B, a corrente do coletor também varie em proporção.

Figura 5 – Conexão de transistor em Emissor-Comum(EC)



Fonte – Retirada de Malvino e Bates (2016).

Segundo afirmam as leis de tensões de Kirchhoff, a soma das tensões numa malha fechada é igual a zero (SCHULER, 2013)(MALVINO; BATES, 2016). Quando aplicada no circuito do transistor (Figura 5), essa a lei estabelece que a tensão entre o coletor e o emissor é igual à tensão na fonte de alimentação menos a tensão no resistor do coletor R_C, resultando na importante fórmula derivada:

$$V_{EC} = V_{CC} - I_C \times R_C, \tag{3.3}$$

Assim, trazendo para o contexto de dissipação de calor, podemos concluir que o transistor gera uma potência igual a tensão coletor-emissor multiplicada pela corrente do coletor, representada por:

$$P_D = V_{EC} \times I_C, \tag{3.4}$$

Com isso, como já foi dito antes, essa potência é a causa do aumento de temperatura do transistor, e o seu valor deve ser comparado com as informações de potência nominal máxima $P_D(mx)$ fornecidas pelas folhas de dados. A dissipação de potência resultante da Equação 3.4 deve ser menor que $P_D(mx)$. Caso contrário, o transistor será danificado.

4 DISPOSITIVO MICROCONTROLADO ARDUINO

4.1 DEFINIÇÃO

O projeto para a implementação do *hardware* e do *software* do Arduino, iniciou-se em 2005, na cidade de Ivrea, Itália, e teve a participação de cinco pesquisadores : Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. Segundo o própio site oficial da plataforma, o Arduino é definido como: "...an open-source electronics platform based on easy-to-use hardware and software." (ARDUINO, 2020b).

O termo *easy-to-use* nesse caso, denota tecnologia de fácil uso e compreensão, sendo assim o ambiente do Arduino foi projetado para ser fácil de usar. De acordo com Michael Margolis (2011), o objetivo seria descomplicar a utilização de uma plataforma de prototipagem eletrônica, por pessoas não técnicas e iniciantes que não necessariamente tivessem experiência em software ou eletrônica. Entretanto, embora esse objetivo tenha sido alcançado, o *hardware* do Arduino funciona no mesmo nível de sofisticação que os engenheiros empregam para criar dispositivos incorporados e complexos. Os profissionais da área e que já trabalham com microcontroladores também foram atraídos para o Arduino devido à sua capacidade de desenvolvimento ágil e à sua instalação para rápida implementação de ideias (MARGOLIS, 2011).

Basicamente, o Arduino possui um microcontrolador que pode ser programado para processar entradas e saídas entre o dispositivo e os componentes externos conectados a ele. Esse tipo de equipamento também é conhecido como plataforma de computação física ou embarcada, ou seja, um sistema que pode comunicar-se com o ambiente por meio de *hardware* e *software*. Nesse aspecto, o Arduino pode ser utilizado para desenvolver projetos interativos e independentes, ou pode ser conectado a um computador que o possibilite recuperar, enviar e atuar sobre um conjunto de dados recebidos de sensores (MCROBERTS, 2015).

4.2 PLATAFORMA DE DESENVOLVIMENTO

A plataforma de desenvolvimento do Arduino é formada por dois componentes principais: *hardware* e *software*. Essa combinação de componentes é o que permite criar projetos que detectam e controlam o mundo físico. O *hardware* é representado por uma placa de prototipagem na qual são construídos os projetos. Já o *software*, chamado de *sketch*, é criado em computador usando o ambiente de desenvolvimento integrado (IDE) do Arduino. Esse IDE permite escrever, editar e converter código em instruções para o hardware do Arduino. O IDE também transfere essas instruções para a placa Arduino em processo chamado *upload* (MARGOLIS, 2011). A seguir, são descritas as funções básicas de ambos os componentes.

4.2.1 *Software*

O ambiente de desenvolvimento integrado (IDE) do Arduino, possui uma linguagem própria de programação baseada na linguagem *Wiring*. Quando iniciado o processo de *upload* da *sketch* para a placa, as rotinas escritas em alto nível de abstração, são traduzidas para linguagem C, e o código é transmitido ao compilador, que por sua vez, realiza a tradução dos comandos para uma linguagem de baixo nível, com instruções mais diretas ao processador e que podem ser compreendidas pelo microcontrolador (ARDUINO, 2020b). A Figura 6 mostra um exemplo de interface do IDE, e suas funções básicas:

Figura 6 – IDE do Arduino, e funções básicas

Fonte – Autor. 1.Verificar(Compilar) 2.Descarregar(Upload) 3.Novo 4.Abrir 5.Salvar 6.Monitor Serial 7.Ambiente de programação 8.Caixa de diálogo 9.Placa do Arduino utilizada

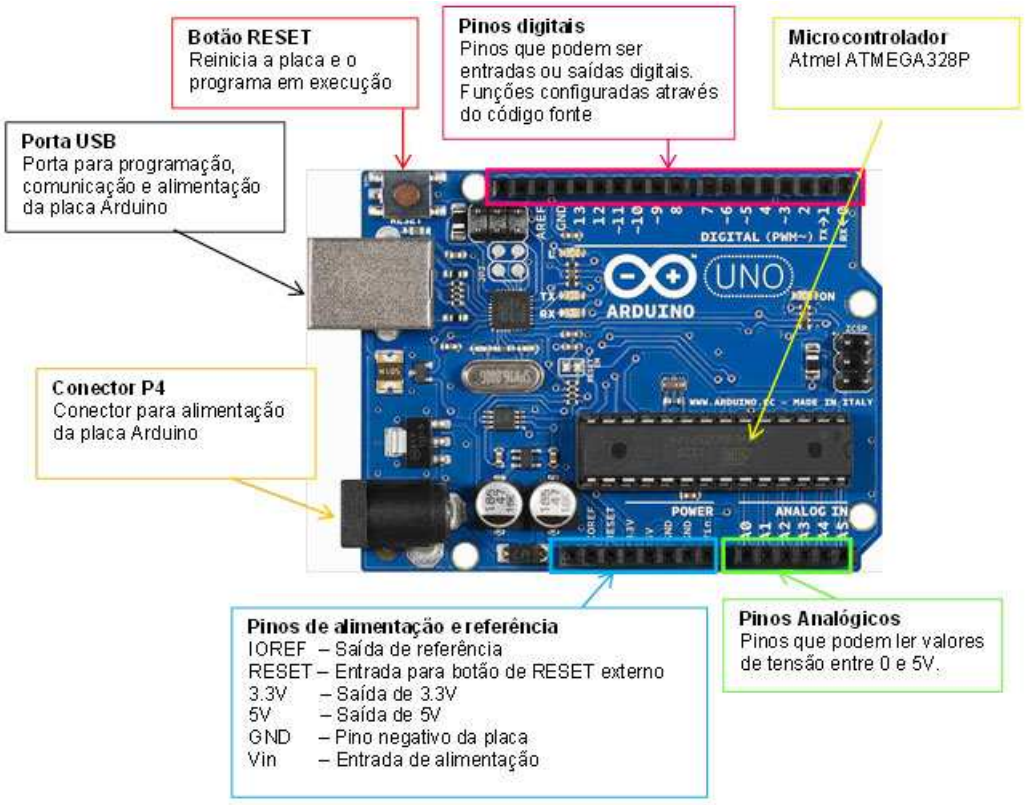
O IDE do Arduino apresenta em suas opções, alguns códigos prontos para serem utilizados como exemplos. O código "*Blink*", mostrado na Figura 6, é relativamente simples e tem como função fazer com que o microcontrolador acione o seu LED integrado através do pino digital 13. As funções *setup()* e *loop()* são obrigatórias, e representam a estrutura básica de qualquer *sketch* do Arduino. A função *setup()* é de configuração, executada apenas uma vez quando o microcontrolador é ligado ou reinicializado, e utilizada por exemplo, para a inicialização de variáveis. Já na função *loop()*, deverá ser escrito todo o código a ser executado continuamente pelo microcontrolador.

No exemplo "*Blink*", está implementado o código que fará o LED integrado do Arduino, piscar em intervalos de tempo de um segundo. A função *digitalWrite(LED_BUILTIN, HIGH)* e *digitalWrite(LED_BUILTIN, LOW)* como o próprio nome sugere, escrevem os valores lógicos *HIGH* ou *LOW* no pino digital do Arduino. Nesse caso, o valor *HIGH*, representa uma tensão de 5V e o valor *LOW* uma tensão de 0V(valores lógicos 1 e 0 respectivamente). Esses valores irão determinar quando ou não o LED deve acender. Por fim, a função *delay*(1000) aguarda o tempo de 1000 milisegundos, para que possa ser executada a próxima instrução do código. Todo o código dentro da função *loop*() é repetido até que o microcontrolador seja desligado ou reiniciado (MARGOLIS, 2011).

4.2.2 *Hardware*

A placa do Arduino é onde o *software* será executado. Esse dispositivo consegue controlar e responder grandezas de eletricidade, de modo que componentes específicos são anexados a ele para permitir a interação com o mundo real. Esses componentes podem ser sensores, que convertem algum aspecto do mundo físico em eletricidade, ou atuadores, que obtêm eletricidade da placa e a convertem em algo para o mundo físico. Exemplos de sensores incluem interruptores, acelerômetros e sensores de temperatura. Atuadores são componentes como luzes e LEDs, alto-falantes, motores e *displays*. Há uma variedade de placas oficiais que podem ser utilizadas com o *software* Arduino e uma ampla variedade de placas compatíveis com Arduino produzidas por membros da comunidade. As placas mais populares contêm um conector *Universal Serial Bus* (USB) usado para fornecer energia e conectividade para carregar o *software* na placa. Os modelos diferem entre si por sua arquitetura e pela finalidade para a qual serão destinados. A Figura 7 mostra a descrição dos principais componentes do modelo Arduino UNO R3, o qual foi utilizado no presente trabalho (ARDUINO, 2020a)(ARDUINO, 2020b).

Figura 7 – Descrição dos terminais e principais componentes da placa Arduino UNO R3.



Fonte – Retirada de página Baú da Eletrônica (2020).

A placa do Arduino UNO é composta sobretudo por um microprocessador de 8 bits Atmel ATMEGA328P, um cristal, que permite sua operação na velocidade correta, enviando pulsos de tempo em uma frequência de 16MHz, um regulador de tensão linear de 5 volts, e uma saída USB, que permite a conexão a um microcomputador para upload e recuperação dos dados (ARDUINO, 2020a).

A alimentação de energia, selecionada automaticamente, é feita via conexão USB ou via fonte de alimentação externa. A energia externa, pode ser proveniente de um adaptador de corrente alternada e contínua (CA-CC) conectado diretamente a placa através de um plugue positivo de 2,1 mm do tipo P4, assim como, pode vir através de uma bateria, com os cabos da mesma inseridos nos pinos de referência e alimentação: GND e Vin. A placa atua com uma fonte externa de 6 a 20 volts. Entretanto, se fornecida alimentação com menos de 7 volts, o pino para a saída de 5V pode fornecer menos tensão do que deveria e a placa pode ficar instável. Da mesma forma, se for utilizado mais de 12 volts, o regulador de tensão pode superaquecer e danificar a placa. Em vista disso, o intervalo recomendado para a fonte externa é de 7 a 12 volts (ARDUINO, 2020a). Os pinos de energia e referência são os seguintes:

- Vin: A tensão de entrada na placa Arduino UNO quando utiliza-se uma fonte de energia externa. A tensão pode ser fornecida diretamente através desse pino, ou acessada, através dele, caso da utilização do adaptador de energia.
- 5V: Esse pino fornece a tensão regulada de 5 volts e pode ser utilizado para a alimentação de *shields* e circuitos externos.
- 3V3. Esse pino fornece a tensão regulada de 3,3 volts e o consumo máximo de corrente é de 50 mA.
- GND: Pino de referência terra.
- IOREF: Esse pino fornece a referência de tensão com a qual o microcontrolador opera. Um dispositivo adequadamente configurado pode ler a tensão do pino IOREF e selecionar a fonte de energia apropriada (5V ou 3,3V).

A placa Arduino UNO possui pinos de entrada e saídas digitais, assim como pinos de entradas e saídas analógicas. Conforme exibido na Figura 7, a placa possui quatorze pinos digitais rotulados de 0 a 13, que operam em 5 volts, e podem fornecer ou receber 20mA como condição operacional recomendada. O máximo de 40mA é o valor que não deve ser excedido em nenhum pino de entrada e saída, a fim de evitar danos permanentes ao microcontrolador. Parte dos pinos digitais do Arduino UNO possuem funções adicionais especiais (ARDUINO, 2020b)(ARDUINO, 2020a), algumas dessas funções são descritas a seguir:

- Serial: Os pinos digitais 0 (RX) e 1 (TX) utilizados para receber comunicação serial. Esses pinos são conectados aos pinos correspondentes do microcontrolador serial ATmega8U2, responsável pela comunicação USB com o PC.
- Interrupções externas: Os pinos 2 e 3. Esses pinos podem ser configurados para acionar uma interrupção em um valor baixo, através da função attachInterrupt().
- PWM: Os pinos 3, 5, 6, 9, 10 e 11 fornecem saída PWM de 8 bits através da função *analogWrite()*.
- LED: Existe um LED integrado a placa do Arduino UNO, e acionado através do pino digital 13. Quando o pino está no valor *HIGH*, o LED acende, e quando o pino está na posição *LOW*, o LED apaga.

Em relação a interface analógica, o Arduino Uno apresenta seis entradas analógicas, rotuladas de A0 a A5, o seu conversor analógico-digital de 10 bits é responsável pela conversão das tensões lidas de 0 a 5 volts, para valores inteiros entre 0 e 1023 (ARDUINO, 2020a).

5 METODOLOGIA

Este trabalho seguiu os preceitos da pesquisa científica aplicada, que conforme Prodanov e Freitas (2013) "objetiva gerar conhecimentos para aplicação prática dirigidos à solução de problemas específicos. Envolve verdades e interesses locais.", e de abordagem quantitativa, na qual as variáveis obtidas podem ser quantificáveis e traduzidas em números.

Para o efetivo desenvolvimento dos objetivos específicos, dividimos as atividades desse projeto em três etapas: Planejamento do Experimento, Implementação do *software*, e por fim, a etapa de Aquisição e análise dos dados.

5.1 AMBIENTE ABERTO PARA EXPERIMENTO DIDÁTICO DE DISSIPAÇÃO DE CALOR

Essa seção descreve o desenvolvimento do ambiente aberto de *hardware* e *software*, baseado no dispositivo microcontrolado Arduino UNO, a fim de favorecer experimentos didáticos de física.

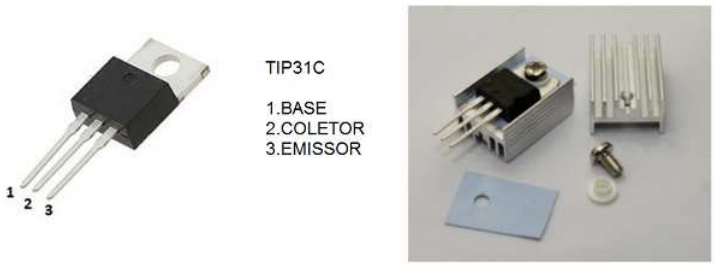
5.1.1 Planejamento do Experimento

O experimento de dissipação de calor, que foi utilizado como prova de conceito para o ambiente aberto de *hardware* e *software* implementado neste projeto, foi elaborado com o propósito de analisar a curva de temperatura de um transistor de junção bipolar polarizado, sob diferentes condições de carga e tensão. As variáveis manipuladas no experimento foram definidas de acordo com as especificações técnicas do componente e têm relação com a temperatura ambiente, o valor dos resistores utilizados e o controle da tensão aplicados ao circuito. A modificação desses valores teve por objetivo identificar mudanças ou não na variável de resposta estudada: a temperatura/dissipação de calor do transistor.

O modelo de transistor escolhido para o experimento foi o transistor bipolar TIP31C. Esse transistor é do tipo NPN de potência, e possui valores elevados quanto as grandezas referentes a tensão e correntes máximas, como também, invólucro e estrutura física facilitadora para a sua montagem junto a um dissipador de calor, qualificando assim, esse componente como

ideal para o experimento em questão. A Figura 8 mostra a pinagem e possível instalação entre um dissipador de calor e o transistor TIP31C.

Figura 8 – Pinos do transistor TIP31 e instalação em dissipador de calor.

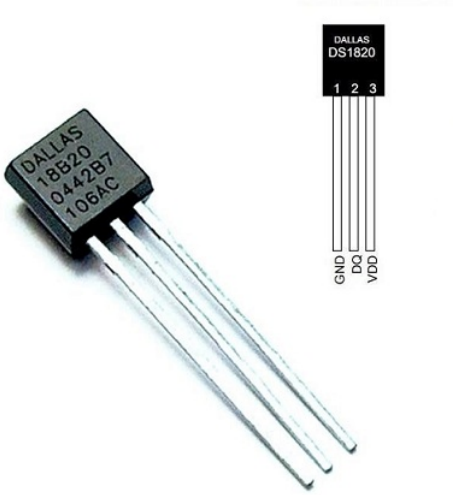


Fonte – Adaptada de página Electrical Engineering (2020).

Por conseguinte o experimento consistiu em utilizar o transistor TIP31C configurado para funcionar nas regiões de operação ativa e saturação. O objetivo foi conseguir com que na medida em que diferentes valores de tensão e corrente fossem aplicados ao circuito, o funcionamento do transistor - ao controlar a corrente que passa entre o seu coletor e o emissor - resultasse em uma dissipação de calor correspondente. Essa dissipação foi medida através de um sensor de temperatura ao longo do tempo de duração do experimento. Esse sensor, através da porta digital do Arduino UNO, envia ao computador os dados de temperatura coletados para que esses possam ser analisados ao final de cada bateria de testes, com a finalidade de identificar o comportamento térmico apresentado pelo transistor.

O sensor de temperatura utilizado junto ao transistor foi o modelo DS18B20 (Figura 9), esse sensor foi escolhido devido a sua precisão e funcionalidade de medição de temperatura com saída diretamente através da porta digital. Além disso, o sensor tem comunicação serial que, por definição, necessita de apenas uma linha de comunicação com o microcontrolador, função conhecida como *One-Wire*, que permite ligar vários sensores de temperatura em uma única entrada digital do Arduino UNO. O intervalo de medidas entre -55^{o}C a +125^{o}C também foi importante característica para a escolha desse sensor, visto que, isso possibilitou uma margem de segurança e maior confiabilidade em relação as altas temperaturas dissipadas pelo transistor (ALFACOMPBRASIL, 2020).

Figura 9 – Sensor de temperatura DS18B20.

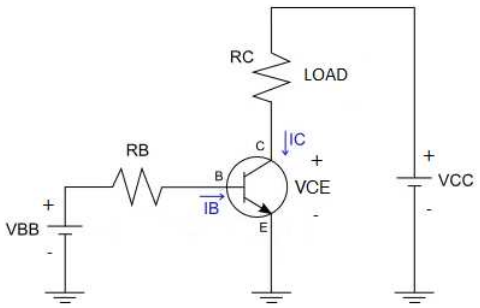


Fonte – Retirada da página FilipiFlop (2020).

5.1.1.1 Circuito de polarização do TIP31C

O circuito utilizado no presente trabalho, representado de forma simplificada pela Figura 10 foi construído, conforme citado anteriormente, para permitir que o transistor TIP31C opere em sua região ativa ou de saturação. Essa característica de operação é necessária, pois, especificamente para o estudo do calor dissipado pelo componente, o transistor polarizado nessas regiões apresenta corrente de coletor I_C e tensão entre coletor e emissor V_{CE} diferentes de zero, portanto, a potência dissipada pelo transistor ($V_{CE} \times I_C$), também é diferente de zero.

Figura 10 – Polarização de transistor em Emissor-Comum(EC)



Fonte – Autor.

Quando o transistor é polarizado na região ativa, uma variação na corrente de entrada do circuito, ou seja, na corrente de base I_B , produz uma variação proporcional na corrente de saída, ou seja, na corrente de coletor I_C. De forma simples, isso significa dizer que a corrente da base, geralmente muito pequena, consegue "controlar"a corrente do coletor que é geralmente muito maior. Isso faz com que o transistor, quando na região ativa, funcione como

um amplificador de sinal. Quando na região de saturação, o transistor funciona como chave fechada permitindo a condução máxima de I_C suportada pelo componente, independente de modificações na corrente de base I_B. A tendência é que o valor de V_{CE} nessa região de operação fique próxima de zero, no entanto na prática, mesmo na saturação a resistência entre coletor e emissor não é nula, o que significa que calor é gerado.

No circuito da Figura 10, o transistor encontra-se na montagem emissor comum, as fontes de alimentação e os resistores polarizam o transistor. Para a correta polarização foi necessário, portanto, estabelecer valores específicos de tensões e correntes nos terminais do transistor, determinando assim o ponto de operação desejado. Os critérios utilizados para a escolha desses valores baseiaram-se na folha de dados do componente disponibilizada na internet[1]. Para o transistor do modelo TIP31C, foi pertinente verificar os valores elétricos relacionados a: dissipação de potência, tensões e correntes máximas. Nesse aspecto, a Tabela 0 mostra os principais parâmetros máximos permitidos para a correta operação do transistor TIP31C.

Tabela 0 – Valores elétricos máximos do TIP31C.

Símbolo	**Parâmetro**	**Valor**	**Unidade**
V_{CBO}	Tensão Coletor-Base ($I_E = 0$)	100	V
V_{CEO}	Tensão Coletor-Emissor ($I_B = 0$)	100	V
V_{EBO}	Tensão Emissor-Base ($I_C = 0$)	5	V
I_C	Corrente do Coletor	3	A
I_{CM}	Corrente de pico do Coletor	5	A
I_B	Corrente de Base	1	A
P_{TOT}	Dissipação Total, com Tcase = 25°C Dissipação Total, com Tamb = 25°C	40 2	W
T_{stg}	Temperatura de armazenamento	-65 a 150	°C
T_j	Máxima temperatura da junção	150	°C

Fonte – Autor.

Importante notar na tabela que os valores correspondentes ao cálculo de potência dissipada são $V_{CEO} = 100$ V e $I_C = 3$ A, porém, ao substituirmos esses valores na fórmula de potência obtivemos um resultado de 300 W, sendo que o valor da potência total do transistor P_{TOT}, na tabela, é de 2 a 40W. Isso acontece por que, para determinar a dissipação de calor no

[1] As folhas de dados podem ser encontradas realizando uma busca simples no navegador com o nome do componente junto a palavra *datasheet*. Para esse trabalho as folhas estavam disponíveis em (ALLDATASHEETS, 2021)

transistor, além da tensão no coletor-emissor e a corrente do coletor devemos também observar a T_{case} e a T_{amb}, esses parâmetros correspondem respectivamente a temperatura do invólucro do transistor e a temperatura ambiente. Ao realizar a construção do circuito para o aparato experimental, essas temperaturas foram métricas importantes a serem observadas, pois, uma variação muito expressiva nos seus valores poderia levar a conclusões errôneas sobre as grandezas elétricas do circuito, assim como, sobres os dados coletados durante o experimento.

5.1.1.2 Parâmetros Experimentais

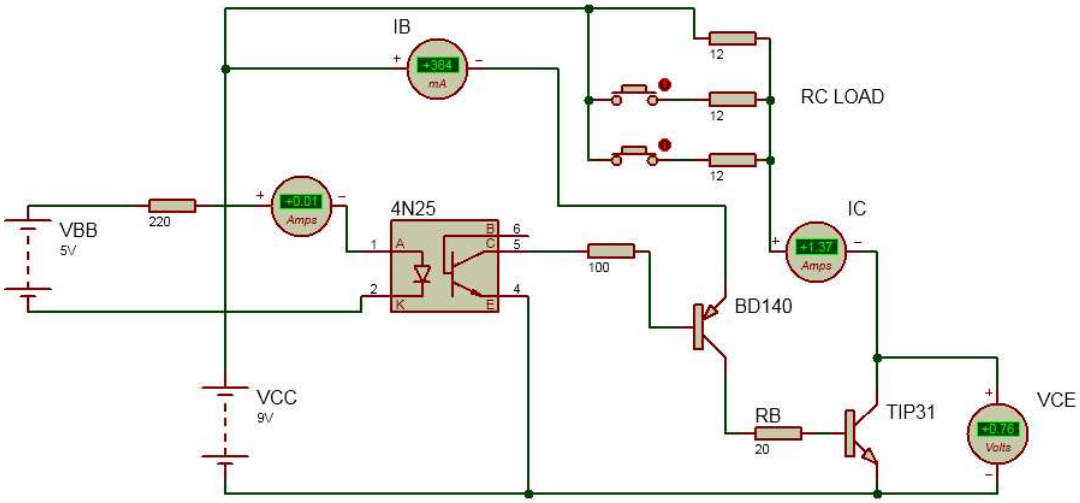
A primeira montagem para o circuito descrito anteriormente, basicamente utilizou o TIP31C para controlar um elemento com resistência R_C LOAD de 12 Ohm, ligado a uma fonte de alimentação V_{CC} de 9V. O controle do elemento foi feito a partir do pico da tensão V_{BB} aplicada na base, que no caso foi de 5 volts, representando a tensão no pino digital do Arduino UNO. Nesse cenário, o elemento funciona quando a tensão aplicada na base, é igual a 5V. O primeiro teste buscou alcançar os parâmetros necessários para que o transistor atuasse na região ativa bem próxima a saturação. Nesse aspecto, verificando a folha de dados do componente constatamos que para atingir esse objetivo os valores necessários seriam: $V_{CE} = 1.2$ V, $I_C = 3$ A, $I_B = 375$ mA.

Esses valores são as condições de teste correspondentes ao funcionamento do transistor na região de saturação. Para atingir essa parametrização, foi preciso a utilização de três resistores R_C LOAD de 12 Ohm ligados em paralelo, assim como, um resistor de 12 Ohm para o resistor da base R_B. Entretanto, analisando os dados das simulações, notamos que apesar dos valores de corrente de base I_B, tensão coletor-emissor V_{CE} e corrente coletor I_C estarem bem próximos ao desejado, esse circuito na prática não seria apropriado. Por questão de segurança e para proteger o Arduino UNO das diferentes tensões aplicadas ao circuito, foi indispensável a utilização do optoacoplador 4N25. Esse componente eletrônico basicamente funciona como uma chave baseada em sinal de luz, ou seja, o que aciona ou não o circuito, é a luz emitida pelo led interno existente no optoacoplador. Essa característica é importante para proteger componentes sensíveis - como é o caso do Arduino UNO - de interfaces com circuitos de maior tensão ou corrente (NOGUEIRA, 2020). Além disso, também verificou-se que as portas digitais do Arduino UNO oferecem uma corrente máxima de apenas 40mA, corrente esta distante do mínimo de 375mA necessário para o circuito em questão. Para resolver esse problema a outra modificação feita no circuito, foi a inclusão de mais um transistor, para que através de uma

ligação do tipo Darlington[2], fosse possível criar uma amplificação do sinal baixo de corrente do microcontrolador na base do transistor TIP31. O transistor escolhido para a ligação Darlington, foi um componente de média potência do tipo PNP, disponível no laboratório durante os testes: o modelo BD140. A Figura 11 representa a esquema final do circuito que foi utilizado no experimento.

Ainda durante a construção física do circuito, foram adicionados botões aos resistores R_C LOAD, a fim de facilitar o acionamento de diferentes valores de resistência durante a execução do experimento. Nesse formato, foi possível aplicar ao circuito, resistências nos valores de 12, 6 e 4 Ohm, dependendo dos botões que eram acionados. O resultado do circuito físico pode ser visto na Figura 12.

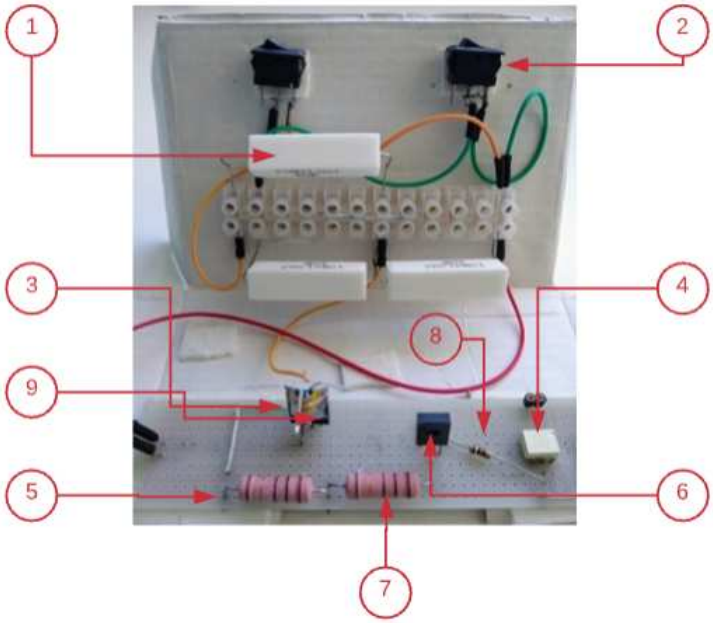
Figura 11 – Esquema final do experimento.



Fonte – Autor.

[2] A ligação Darlington atua como um transistor composto, com um ganho de corrente que é o produto dos ganhos individuais de cada transistor.

Figura 12 – Circuito físico do experimento.



Fonte – Autor. 1.Resistores Rc LOAD(12 Ohm) 2.Botão 3.Sensor DS18B20 4.Optoacoplador 4N25 5.Resistor(220 ohm) 6.Transistor BD140 7.Resistor RB(20 Ohm) 8.Resistor(100 ohm) 9.Transistor TIP31C

Sendo assim, após finalizado o planejamento do experimento, seguimos para a implementação do *software* responsável por coletar e gerenciar os dados de temperatura, essa etapa será descrita na próxima seção.

5.1.2 Implementação do *software*

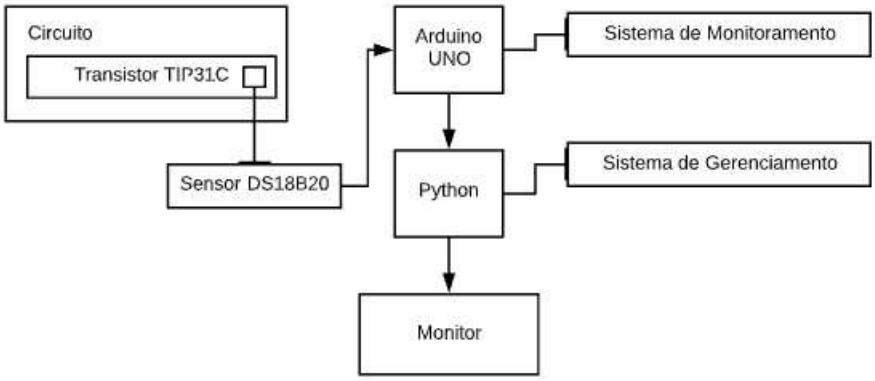
O ambiente aberto para experimentos didáticos de física, proposto nesse projeto, é composto por dois sistemas que se comunicam entre si. O Sistema de Monitoramento implementado na linguagem de programação C na própria IDE do Arduino UNO, e o Sistema de Gerenciamento implementado na linguagem Python (PYTHON, 2021).

Após a correta configuração do circuito do transistor TIP31C descrito na seção anterior, o próximo passo foi conectar o mesmo ao Arduino UNO. O *software* desenvolvido nessa fase foi o Sistema de Monitoramento (SM), que tem como objetivos controlar essa conexão, coletar as temperaturas captadas pelo sensor DS18B20 e enviá-las via porta serial ao Sistema de Gerenciamento (SG).

A Figura 13 representa uma simplificação do ambiente que foi construído. O sensor DS18B20 (fixado diretamente ao transistor TIP31C, com o auxílio de pasta térmica) captura por condução térmica, os dados de temperatura dissipada pelo TIP31C ao longo do tempo do

experimento. O recebimento desses dados é tratado pelo SM que por sua vez os envia ao SG, no qual, toda a informação é organizada e exibida no monitor. O código-fonte do SM é descrito na próxima seção.

Figura 13 – Arquitetura do ambiente de *hardware* e *software*.



Fonte – Autor.

5.1.2.1 Desenvolvimento do Sistema de Monitoramento

A empresa Dallas Semiconductor, fabricante do sensor de temperatura DS18B20, como citado anteriormente utiliza o protocolo de comunicação *One-Wire*. As primeiras linhas de código do SM referem-se à inclusão das bibliotecas *OneWire.h* e *DallasTemperature.h* necessárias para a utilização desse protocolo. Após essa inclusão, a variável constante PIN_SENSOR do tipo inteiro é criada para definir em qual pino digital do Arduino UNO o sensor será conectado. Essa variável é utilizada também para instanciar o objeto do tipo oneWire que será, na próxima instrução, referenciado pela biblioteca DallasTemperature.h. Ao final desse trecho do Código-fonte 1 é criada a variável PIN_TIP31 do tipo inteiro, que armazena o pino digital no qual o circuito do transistor TIP31C será conectado.

Código-fonte 1 – SM - Inclusão de bibliotecas e definição dos pinos

```
1 #include <OneWire.h>
2 #include <DallasTemperature.h>
3
4 const int PIN_SENSOR = 2;
5 OneWire oneWire(PIN_SENSOR);
6 DallasTemperature sensors(&oneWire);
7
```

```
8 int PIN_TIP31 = 11;
```

Fonte – Autor.

As próximas linhas do código abordam o bloco de configuração obrigatório de qualquer programa do Arduino: a função *void setup()*.

Como pode ser visto na linha 8 do Código-fonte 1 o pino digital utilizado pelo circuito do TIP31C é o pino 11. Para determinar o estado ligado ou desligado do circuito, o Arduino UNO precisa enviar sinais de tensão através desse pino, é por esse motivo, que utilizando a função *pinMode()*, o PIN_TIP31 é configurado como um pino de saída (OUTPUT). É preciso executar essa instrução para declará-lo explicitamente, visto que normalmente os pinos do Arduino UNO operam como entrada. Na sequência, o método *Serial.begin()*, abre a porta para comunicação serial com o computador e define a taxa de transmissão dos dados em bits por segundo. Como pode ser visto no Código-fonte 2 a taxa de transmissão escolhida foi de 9600 bps.

Para fechar o bloco *void setup()* a instrução *sensors.begin()* inicia os sensores de temperatura conectados. Para esse projeto apenas um sensor foi utilizado.

Código-fonte 2 – SM - Função de configuração *void setup()*

```
1 void setup(void)
2 {
3   pinMode(PIN_TIP31, OUTPUT);
4   Serial.begin(9600);
5   sensors.begin();
6 }
```

Fonte – Autor.

Outro bloco de função obrigatório para programas implementados na IDE do Arduino é o *void loop()*. Dentro dessa função geralmente está o código principal do programa, que será executado constantemente até que o microcontrolador seja desligado ou reiniciado.

Para a implementação em questão, no *void loop()* é definida a lógica de comunicação com o software em Python: o SG.

Durante a comunicação, uma variável do tipo char byteEntrada é criada para armazenar o *byte* que o SG envia através da porta serial. A lógica é verificar se o numero de *bytes* na porta serial é maior que zero, utilizando a instrução *Serial.available()* > 0. Após verificação verdadeira, é realizada a leitura do *byte* e respectivo armazenamento na variável. Em seguida, é feita uma comparação entre o *byte* recebido e os valores definidos em instrução de controle *switch*. Como pode ser verificado no Código-fonte 3, a depender do *byte* recebido o controle de tensão aplicado sobre o circuito do TIP31C é diferente. A instrução *analogWrite(PIN_TIP31, 63)* por exemplo, permite simular no PIN_TIP31 um sinal analógico de nível 63. Na prática, isso significa que o sinal lógico do pino alterna entre os níveis de tensão alto e baixo. O nível é o valor conhecido por *Duty Cycle* que varia de 0 a 255 e determina a parcela de tempo em que o sinal ficará em nível alto. No exemplo, como 255 equivale teoricamente a 100%, o nível 63 significa dizer que o PIN_TIP31 ficará em nível alto de tensão em aproximadamente 25% do tempo.

Código-fonte 3 – SM - Função *void loop()*

```
1  void loop(void)
2  {
3    char byteEntrada;
4    if (Serial.available()>0){
5      byteEntrada = Serial.read();
6
7      switch(byteEntrada){
8        case "A":
9          analogWrite(PIN_TIP31, 63);
10         break;
11       case "B":
12         analogWrite(PIN_TIP31, 127);
13         break;
14       case "C":
15         digitalWrite(PIN_TIP31, HIGH);
```

```
16          break;
17        case "S":
18          digitalWrite(PIN_TIP31, LOW);
19          break;
20      }
21    }
22      sensors.requestTemperatures();
23      Serial.println(sensors.getTempCByIndex(0));
24      delay(1000);
25  }
```

Fonte – Autor.

Em relação a instrução *digitalWrite()*, os valores *LOW* e *HIGH* parametrizados para essa função, representam respectivamente sinais baixo e alto de tensão constantes iguais a 0 e 5 volts. Dessa forma, sinais diferentes de tensão podem ser enviados ao circuito, e as funções *analogWrite()* e *digitalWrite()* são as responsáveis por determinar o controle desse sinal.

Finalizando a implementação do SM, a últimas linhas de código referem-se à requisição das temperaturas capturadas pelo sensor DS18B20 e subsequente impressão dos dados no monitor serial do IDE. Importante notar o método *.getTempCByIndex(0)* que determina simultaneamente a conversão da temperatura em graus Celsius, e o índice do sensor selecionado. Para esse projeto, como foi utilizado apenas um sensor de temperatura, o índice é igual a zero.

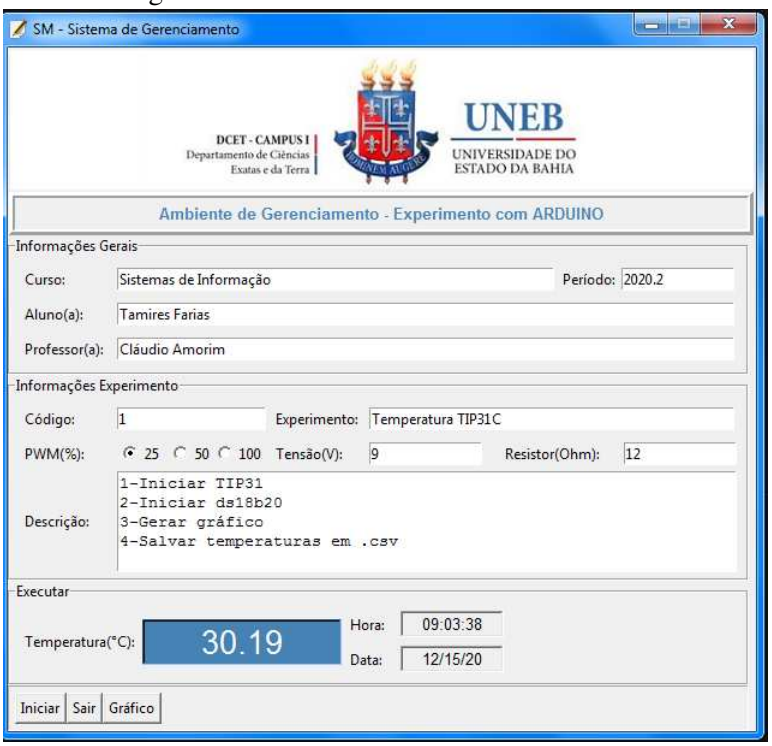
Após o código descrito anteriormente ser corretamente compilado e carregado para o Arduino UNO, a próxima etapa de desenvolvimento consistiu em implementar, utilizando a linguagem Python, o *software* SG.

5.1.2.2 Desenvolvimento do Sistema de Gerenciamento

O SG foi desenvolvido com o objetivo de controlar a execução do experimento. Por meio desse *software* é possível iniciar e interromper o experimento, como também, visualizar e salvar os dados de temperatura coletados. Na tela inicial, conforme a Figura 14, devem ser informadas as informações gerais (nome do aluno, professor, período letivo etc.) assim como, as informações do experimento (controle de tensão, Tensão(V), Resistor(Ohm) etc.). Esses registros

são gravados junto com os dados de temperatura, em arquivo no formato .csv, no decorrer da execução do experimento. Os botões localizados no canto inferior esquerdo da tela, funcionam respectivamente para iniciar, sair do experimento e plotar o gráfico dos dados recebidos. A partir do momento em que o botão "Iniciar"é acionado, o SG exerce controle sobre o andamento do experimento e começa a compartilhar informações com o SM. Basicamente, o SG envia o controle de tensão necessário para iniciar o circuito do transistor TIP31C e o SM, após aplicar a tensão desejada, retorna as temperaturas dissipadas pelo transistor ao longo do tempo. Esse processo ocorre até que uma temperatura máxima (previamente parametrizada) seja alcançada. Todo procedimento é observado em tempo real. As informações essenciais para o entendimento dessa implementação serão descritas a seguir.

Figura 14 – Sistema de Gerenciamento - SG.



Fonte – Autor.

Ao trabalhar com plataforma de *hardware*, é fundamental ter algum meio de comunicação entre o *hardware* e o computador que é utilizado para a programação. Entre os procedimentos comuns de interface entre computador e *hardware*, a comunicação baseada em porta serial USB é uma das mais usuais e simples de estabelecer, especialmente para a plataforma Arduino. Nesse aspecto, a linguagem Python foi escolhida para a elaboração do SG, pois, fornece uma biblioteca denominada *pySerial* que lida com esse tipo de comunicação, é de fácil entendimento e rápida implementação (DESAI, 2015). Ademais, os recursos adicionais

para interface gráfica de usuário[3], gráficos para visualização de dados e armazenamento de dados foram determinantes para a escolha do Python para este projeto, visto que, a linguagem tem várias outras bibliotecas, de código aberto e amplamente suportadas pela comunidade, que auxiliam o desenvolvimento de cada um dos mecanismos mencionados anteriormente.

A primeira fase de desenvolvimento do SG, teve o objetivo de fornecer a interface entre o ambiente de programação Windows e o Arduino UNO. Nessa etapa, a utilização da biblioteca *pySerial* é o que permite a comunicação com a plataforma de *hardware*, encapsulando o acesso e configuração da porta serial diretamente por meio do interpretador Python. Após a correta instalação da biblioteca, a implementação fundamental para a *pySerial* consistiu em apenas três linhas de código.

Código-fonte 4 – SG - Utilização da biblioteca *pySerial*

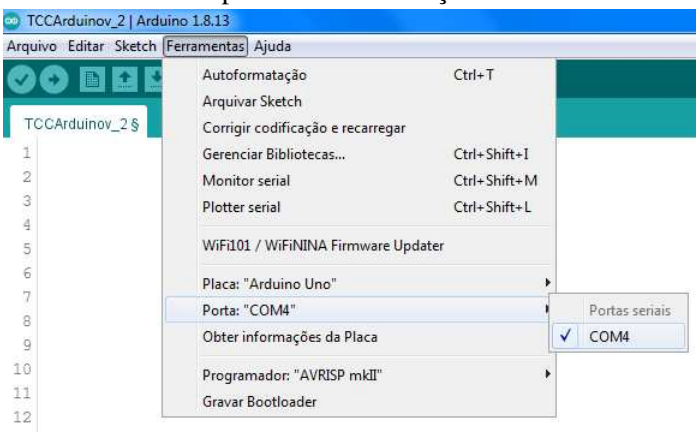
```
1 import serial
2 ser = serial.Serial("COM4", 9600, timeout=1)
3 ser_bytes = ser.readline()
```

Fonte – Autor.

Conforme o Código-fonte 4, para utilizar a biblioteca é necessário importar o módulo utilizando a instrução *import serial*. Depois, com a chamada ao método *.Serial()*, é determinada a configuração para abrir a conexão com a porta. Para verificar em qual porta serial o Arduino UNO está conectado, no Windows o processo é simples. Quando a placa é conectada pela primeira vez, o sistema operacional instala automaticamente os drivers necessários, e logo que esse processo é concluído, é possível na IDE do Arduino navegar na barra de menus até Ferramentas | Porta e selecionar a porta disponível (Figura 15). No código em questão, a porta 'COM4' foi definida para a comunicação a uma taxa de transferência de 9600 bits por segundo e *timeout* no valor de 1 segundo. A leitura dos dados enviados pelo Arduino UNO é feita pelo método *.readline()* que lê as informações até que encontre o caractere de nova linha "\n".

[3] Foi utilizada para este projeto, a biblioteca de interface gráfica do usuário *Tkinter*. Essa biblioteca faz parte do pacote de instalação de todas as versões do Python.

Figura 15 – Selecionar a porta de comunicação serial - IDE ARDUINO.



Fonte – Autor.

A fim de conseguir a leitura de todos os registros de dados em tempo real, foi necessário a implementação de uma estrutura de repetição *while*. A partir disso, na medida em que os dados de temperatura eram lidos, também foi identificada a necessidade de conversão para o formato *Unicode*, uma vez que, o Arduino UNO envia strings de *bytes* e comandos ASCII. O Código-fonte 5 é um trecho simplificado [4] da principal estrutura *while* do SG, responsável por ler, decodificar e gravar em arquivo .csv os dados de temperatura do transistor TIP31C.

Código-fonte 5 – SG - Loop de leitura serial das temperaturas

```
1  import serial
2
3  tempmax = 80.0
4  ser = serial.Serial("COM4", 9600, timeout=1)
5  name= time.strftime("%Y-%m-%d_%H%M%S")
6  filename = "EXP_%s.csv"% (name)
7
8  while (decod_bytes <= tempmax):
9      ser_bytes = ser.readline()
10     try:
11         decod_bytes = float(ser_bytes[0:len(ser_bytes)-2].
               decode("utf-8"))
12         print(decod_bytes)
```

[4] Para facilitar a leitura e o entendimento, foram excluídas várias linhas de código referente à interface gráfica do usuário. O código completo está disponível no apêndice C.

```
13          with open(filename,"a", newline="") as arquivo_csv:
14              writer = csv.writer(arquivo_csv, delimiter=";")
15              writer.writerow([time.strftime("%H:%M:%S"),
                    decod_bytes])
16      except:
17          continue
18      ser.flushInput()
19  else:
20      ser.write(b"S")
21      break
```

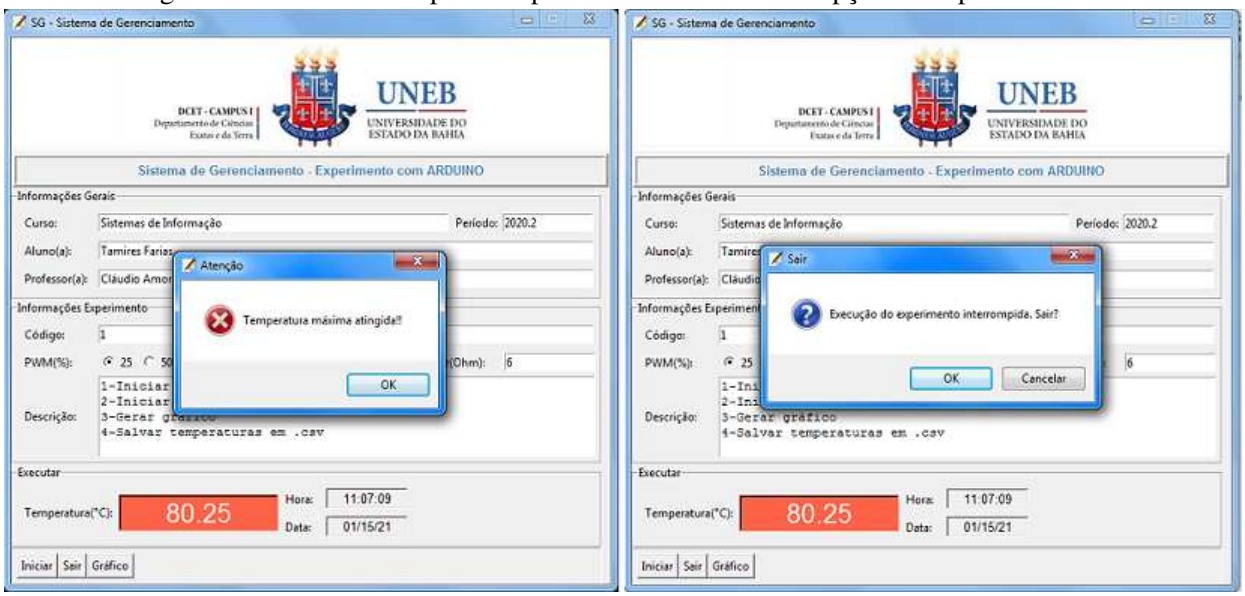
Fonte – Autor.

Os dados de temperatura são lidos continuamente até que a condição de parada seja verdadeira. A variável global tempMax na linha 3, armazena a temperatura máxima a qual o transistor pode atingir durante a execução do experimento. No trecho de código mencionado, essa temperatura foi definida como 80.0 (oitenta graus Celsius). Por conseguinte, a leitura dos dados ocorrerá até que a condição de parada *(decod_bytes <= tempMax)* seja verdadeira. A variável *decod_bytes* é responsável por armazenar o resultado da conversão dos bytes recebidos através da comunicação serial. O formato de dados recebido pelo Python, por exemplo, para a temperatura igual a 80.0 é algo como: b'80.00\r\n'. Então, para que o SG consiga processar esse dado corretamente, é essencial a utilização do método *.decode("utf-8")* para interpretar a string usando a codificação fornecida. Nesse caso, o "utf-8"foi escolhido por ser um formato de codificação de caracteres comumente usado.

A Figura 16 mostra a ação do sistema quando a temperatura máxima é atingida. Após informar ao usuário a ocorrência do evento, o SG executa o método *.write(b"S")*, que transmite via comunicação serial, o *byte* "S"ao Sistema de Monitoramento. Por sua vez, o SM interpreta o *byte* como a instrução que envia o sinal de tensão de 0 volts ao circuito. Consequentemente o experimento é interrompido e finalizado, a fim de preservar a estrutura elétrica do transistor TIP31C.

Simultaneamente aos processos descritos anteriormente, existe a escrita do arquivo .csv. O modo de arquivo "a", como visto na linha 14 do Código-fonte 5, anexa novas linhas ao

Figura 16 – SG - Alertas para temperatura máxima e interrupção do experimento.



Fonte – Autor.

final do arquivo existente, e o argumento "*newline*"impede que espaços vazios sejam erroneamente gravados. Sendo assim, além das informações preenchidas pelo usuário, são gravadas a data, hora e temperaturas coletadas. Finalmente, o nome do arquivo é padronizado de acordo com a data e hora inicial do experimento, no seguinte formato: EXP_[data]_[hora]. A Figura 17 mostra um exemplo de arquivo gravado após a execução do experimento.

Figura 17 – Arquivo .csv gerado pelo SG.

1	Arduino/Dados Porta Serial	
2	Curso:	Sistemas de Informação
3	Período:	2020.2
4	Aluno:	Tamires Farias
5	Professor:	Cláudio Amorim
6	ID :	1
7	Experimento:	Temperatura TIP31
8	Tensão(V):	9,0
9	Resistor(Ohm):	12
10	PWM(%):	100
		1-Iniciar TIP31
		2-Iniciar ds18b20
		3-Gerar gráfico
11	Descrição:	4-Salvar temperaturas em .csv
12		
13	Data:	12/17/20
14	Hora	Temperatura °C
15	10:30:48	32
16	10:30:50	34.13
17	10:30:51	34.25
18	10:30:53	34.38
19	10:30:54	34.56
20	10:30:55	34.81
21	10:30:57	35.19
22	10:30:58	35.25
23	10:31:00	35.25

EXP_2020-12-17_103032

Fonte – Autor.

Com a utilização do SG também existe a possibilidade de plotar um gráfico de linha na medida em que as temperaturas são recebidas. Através do botão "Gráfico"o SG abre outra

janela na qual a plotagem em tempo real dos dados é exibida. O Código-fonte 6 mostra as principais instruções dessa implementação.

Código-fonte 6 – SG - Loop para plotagem do gráfico de temperaturas

```
import matplotlib
matplotlib.use("tkAgg")
import matplotlib.pyplot as plt
import numpy as np

plt.figure(num = "Grafico - Temperaturas")
plt.ion()
data = np.array([])

while (decod_bytes <= tempmax):
    ser_bytes = ser.readline()
    try:
        decod_bytes = float(ser_bytes[0:len(ser_bytes)-2].
           decode("utf-8"))
        print(decod_bytes)
        with open(filename,"a", newline="") as arquivo_csv:
            writer = csv.writer(arquivo_csv, delimiter=";")
            writer.writerow([time.strftime("%H:%M:%S"),
               decod_bytes])
    except:
        continue
    plt.title ("Temperatura transistor TIP31C")
    plt.xlabel("Tempo (s)")
    plt.ylabel("Temperatura C")
    plt.grid(True)
    data = np.append(data, decod_bytes)
    plt.plot(data, "b-")
    plt.pause(0.01)
```

```
27 else:
28     ser.write(b"S")
29     break
```

Fonte – Autor.

As primeiras linhas do Código-fonte 6 relacionam-se com as importações necessárias para a utilização da biblioteca de plotagem Python: matplotlib. Essa biblioteca é amplamente usada em aplicativos baseados em Python para visualização e análise de dados, e fornece, através da utilização do *framework* pyplot, diversas funções e métodos para traçar e configurar gráficos. Além disso, geralmente a matplotlib utiliza as funções de cálculos numéricos de outra biblioteca do Python: Numpy.

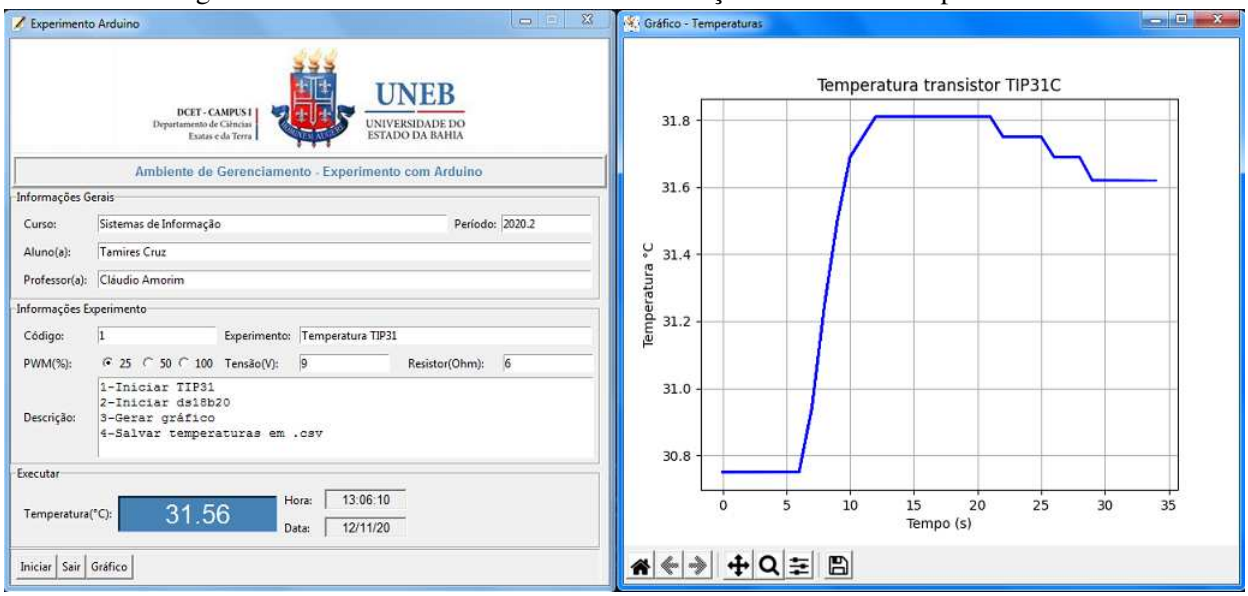
O primeiro passo para a exibição do gráfico consiste em determinar a figura na qual esse gráfico será desenhado. Na linha 6 do Código-fonte 6 o método *.figure()* cria e nomeia essa estrutura. Na sequência, é preciso criar o modo interativo através do método *.ion()*, no intuito de permitir que o gráfico, na medida em que os dados de temperatura sejam recebidos via porta serial, altere dinamicamente os seus eixos x e y.

A lógica de recebimento dos dados é a mesma explicada anteriormente, a diferença é que desta vez, os dados após correta decodificação para tipo *float*, são armazenados na matriz data (linha 24). Por conseguinte, o método *.plot ()* recebe essa estrutura de dados como entrada e traça o gráfico. Como apenas a matriz data foi fornecida, o *.plot()* assume que os valores são a sequência utilizada para o eixo y e usa valores incrementais gerados automaticamente para o eixo x. O gráfico por fim, é atualizado a cada segundo.

Os outros métodos exibidos no Código-fonte 6 são referentes ao design do gráfico, com eles é possível mudar o título do gráfico, os títulos dos eixos e exibir uma grid para melhor observação. A Figura 18 traz a visualização do SG, quando o gráfico de temperaturas é exibido.

Finalizada a etapa de implementação do *software*, seguimos a fase final do projeto na qual o experimento foi efetivamente executado. Na próxima sessão são explicadas como a aquisição de dados e posterior análise foi realizada.

Figura 18 – Sistema de Gerenciamento - Exibição Gráfico de Temperaturas



Fonte – Autor.

5.1.3 Aquisição e Análise dos Dados

Nesta etapa foi iniciada a bateria de experimentos. Durante a realização dos mesmos, as variáveis de entrada foram modificadas de forma iterativa, a fim de observar possíveis alterações nos resultados. O objetivo aqui, foi comprovar o uso do ambiente de *hardware* e *software* como um facilitador na manipulação e observação dos dados envolvidos no processo didático da experiência.

O experimento de dissipação de calor, utilizado como prova de conceito para o ambiente de *hardware* e *software* implementado neste projeto, foi elaborado com o propósito de analisar a curva de temperatura do transistor TIP31C, sob diferentes condições. As variáveis de entrada para esse procedimento, conforme a Tabela 1, foram definidas de acordo com as especificações técnicas do transistor e têm relação com a tensão, temperatura ambiente, condição de ventilação e o valor da resistência R_C LOAD no circuito. A alteração dessas variáveis, na realização de cada experimentação, teve como objetivo identificar ou não mudanças na variável de resposta estudada: a temperatura/dissipação de calor do transistor TIP31C.

No total foram realizados seis experimentos. Antes do início de cada ensaio, foram definidas as variáveis de entrada mediante a medição da temperatura ambiente através de um multímetro[5] digital, equipado com sensor de temperatura, aplicação ou não da ventilação forçada e direta sobre o circuito e acionamento dos resistores R_C LOAD. Como citado em seção anterior,

[5] Equipamento eletrônico que tem por função medir algumas grandezas elétricas.

Tabela 1 – Variáveis de entrada.

Experimento	**Tensão Vbb (V)**	**Tensão Vcc (V)**	**Temperatura Ambiente (°C)**	**Ventilação**	**Resistência RC LOAD (Ohm)**
1	5	9	30	Sim	12
2			31	Não	12
3			30	Não	4
4			30	Sim	4
5			30	Não	6
6			31	Sim	6

Fonte – Autor.

os resistores foram ativados através de botões do circuito, fornecendo valores de resistências iguais a 4 Ohm, 6 Ohm ou 12 Ohm. As tensões V_{BB} e V_{CC} não foram modificadas durante todos os experimentos e mantiveram-se respectivamente nos valores de 5V e 9V.

A experimentação foi realizada no período de dois dias. Cada experimento gerou um conjunto de resultados exibidos e organizados conforme a Tabela 2. Essa tabela relaciona, a tensão V_{CE}, a corrente de coletor I_C, potência dissipada P_D, média das temperaturas, amplitude das temperaturas, temperatura mínima e máxima e intervalo de temperatura de maior frequência. A duração de cada experimento não teve um valor fixo, e foi determinada de acordo com a visualização em tempo real dos dados e eventual estabilidade das temperaturas captadas.

Tabela 2 – Conjunto de resultados.

Exp.	**Vce (V)**	**Ic (A)**	**Pd (W)**	**Média Temperaturas (°C)**	**Amplitude Temperaturas (°C)**	**Temperatura Mín-Máx (°C)**	**Intervalo maior Frequência**
1	0.6	0.7	0.4	40.2	9.9	32.0 - 41.9	[40.0 ; 41.9[
2	0.6	0.7	0.4	54.4	30.5	30.4 - 60.9	[57.4 ; 60.4[
3	1.7	1.8	3	66.7	48.9	27.9 - 76.8	[69.9 ; 75.9[
4	1.7	1.8	3	55	31.0	27.9 - 58.9	[55.9 ; 59.9[
5	1.3	1.2	1.5	60.5	39.0	28.6 - 67.6	[64.6 ; 68.6[
6	1.3	1.2	1.5	45.2	21.3	27.4 - 48.7	[45.4 ; 48.4[

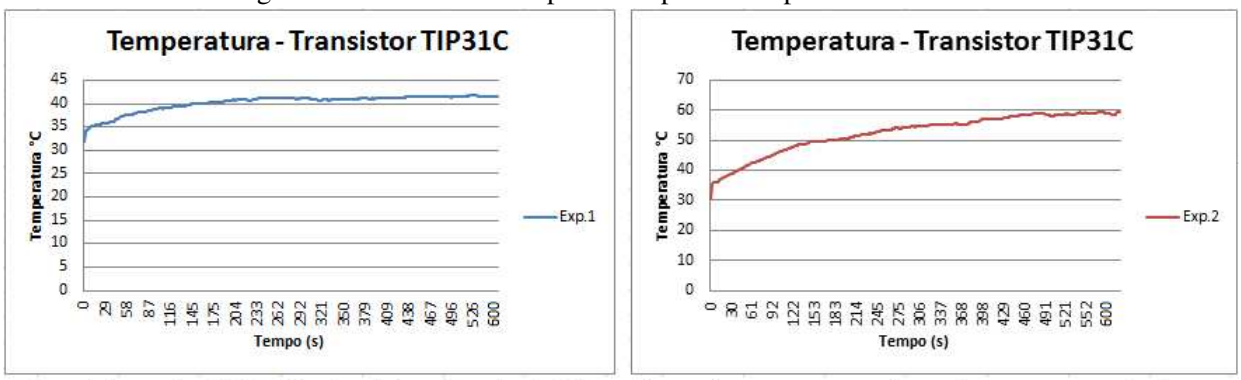
Fonte – Autor.

Para o cenário do Experimento 1, conforme a Tabela 1, a temperatura ambiente foi de 30°C. Houve a utilização de ventilação forçada sobre o circuito e a resistência R_C LOAD aplicada teve valor igual a 12 Ohm. Existiu uma dificuldade em medir a tensão entre o coletor e emissor do transistor, devido a montagem física do aparato experimental. Por esse motivo, a

tensão V_{CE} foi calculada de acordo com a tensão sobre os resistores R_C LOAD. Desse modo, com a utilização do multímetro, a tensão mensurada sobre R_C LOAD, para o Experimento 1, resultou no valor de 8.4V. Consequentemente a tensão restante sobre o transistor TIP31C foi igual a 0.6V, visto que a V_{CC} aplicada ao circuito foi igual a 9V. O cálculo da corrente no coletor do transistor utilizando Lei de Ohm, resultou em 0.7A.

Com esse valores explorados, foi possível finalmente, por meio do botão "Iniciar"do SG, começar o experimento e verificar as temperaturas captadas pelo sensor DS18B20, assim como, a curva de temperatura traçada. A Figura 19 mostra em (a) o gráfico da curva de temperatura do TIP31C pra o Experimento 1, e em (b) o gráfico gerado para o cenário do Experimento 2, similar ao anterior descrito, mas sem a utilização de ventilação forçada sobre o transistor.

Figura 19 – Gráfico de temperaturas para os Experimentos 1 e 2.



(a) Curva de temperatura TIP31C do Experimento 1, com tempo igual a 600 segundos, RC LOAD de 12 Ohm e ventilação forçada sobre o circuito.

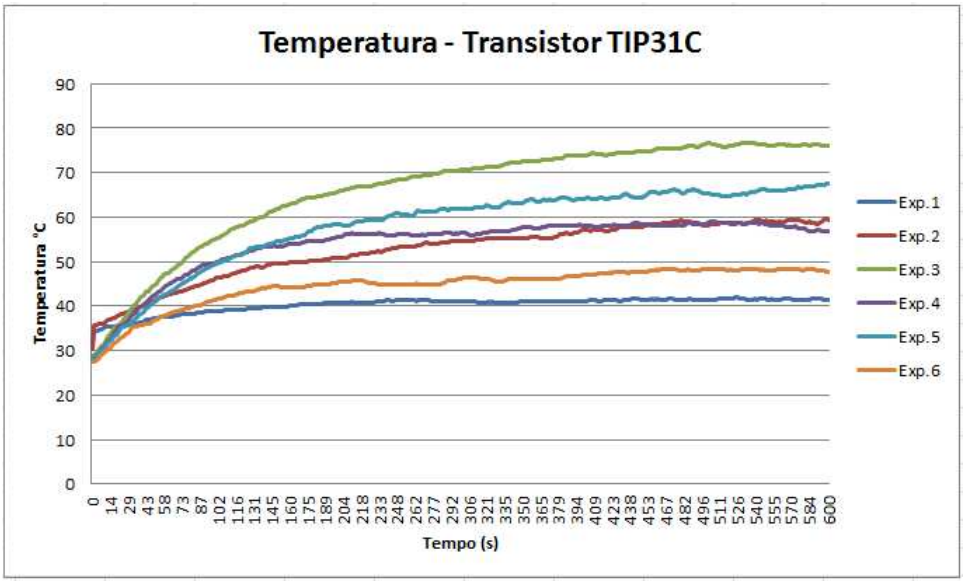
(b) Curva de temperatura TIP31C do Experimento 2, com tempo igual a 600 segundos, RC LOAD de 12 Ohm e sem ventilação forçada sobre o circuito.

Fonte – Autor.

É possível identificar observando os gráficos uma expressiva alteração na amplitude das temperaturas capturadas. Apenas modificando a utilização ou não da ventilação sobre o circuito, enquanto no Experimento 1 a amplitude de temperatura chegou próximo aos 10 °C, no Experimento 2 (sem ventilação) essa diferença chegou a aproximadamente 30 °C. Durante todo o processo de experimentação, não foi possível um controle rigoroso quanto a regulagem de temperatura ambiente e de umidade do ar, duas grandezas que interferem diretamente na dissipação de calor. Apesar disso, foi nítida a sensibilidade do transistor em relação a circulação de ar no ambiente. Ao observar a plotagem do gráfico em tempo real, mudanças instantâneas foram identificadas a qualquer mínima alteração quanto a ventilação local. A abertura de uma janela, utilização de ventilação forçada e até mesmo a aproximação das mãos ao circuito eram suficientes para interferir nas temperaturas captadas.

A temperatura máxima de 80°C foi parametrizada como o ponto de interrupção para os experimentos. Essa temperatura não foi atingida em nenhum dos seis ensaios, porém, o Experimento 3, utilizando resistência R_C LOAD de 4 Ohm e sem ventilação sobre o circuito, foi o que obteve resultados mais próximos a temperatura máxima, chegando a atingir valores na faixa de 76°C, resultando, conforme a Tabela 2, numa potência dissipada de 3W. Temperaturas altas também foram atingidas quando utilizada resistência R_C LOAD de 6 Ohm. No Experimento 5, com o circuito sem ventilação e temperatura ambiente em 30°C, a temperatura máxima alcançada foi de 67.6°C, com a corrente de coletor 1.2A, tensão entre o coletor e emissor 1.3V e potência dissipada 1.5W. A Figura 20 mostra o gráfico para todos os experimentos realizados.

Figura 20 – Gráfico de temperaturas para os Experimentos 1 a 6.



Fonte – Autor.

Importante salientar, que durante a experimentação houve inconsistências nas medições de tensões do circuito. Os resistores R_C LOAD apesar de teoricamente iguais, apresentaram diferentes valores de tensão mesmo estando configurados em paralelo. Após rápida verificação, constatou-se que os botões e fios instalados nas ligações dos resistores não apresentavam resistência nula. Essa baixa resistência resultou em queda de tensão nesses botões, de modo que a medição de tensão em cada resistor ficou ligeiramente diferente. Contudo, essa diferença não afetou os cálculos de corrente I_C sobre o transistor. A variação de corrente utilizando um, dois ou três resistores (que equivale respectivamente a 12 Ohm, 6 Ohm e 4 Ohm) ficou dentro do nível de precisão desejável. Além disso, os valores ficaram relativamente próximos aos simulados antes da construção física do circuito. Sendo assim, a diferença na medição de tensão entre os

resistores R_C LOAD, foi desconsiderada.

Conforme demonstrado anteriormente, a partir da observação em tempo real, e subsequente análise dos dados, pôde-se inferir algumas informações acerca do comportamento térmico do transistor TIP31C. O fato de todos os dados serem salvos em arquivo possibilitou e facilitou os cálculos posteriores e a correção de possíveis erros. Mediante a organização e gravação dos dados foi possível por exemplo, corrigir um erro quanto a precisão das temperaturas coletadas. Apesar da realização de apenas seis experimentos, o volume de temperaturas foi relativamente grande, 3069 no total. Todas essas temperaturas foram gravadas com a utilização da segunda casa decimal. Depois, verificou-se que o arrendondamento desses valores era necessário, devido a precisão de $\pm 0.5\,^{\circ}\mathrm{C}$ fornecida pelo sensor DS18B20. Nesse caso, foi possível mesmo depois de todos os experimentos finalizados, manipular o arquivo .csv para que esse erro fosse corrigido, demonstrando assim a utilidade na automatização de todo o processo.

Para validar o ambiente livre de *hardware* e *software* implementado neste projeto, como um equipamento potencialmente significativo para a aprendizagem, inicialmente além da verificação quantitativa dos dados, seria realizada uma validação qualitativa do ambiente. Essa validação seria feita por meio da aplicação de entrevista estruturada com especialistas da área, no caso, um grupo de professores do curso de Licenciatura em Física da UNEB - DCET, Campus Salvador. Entretanto, devido as condições de isolamento social impostas pela pandemia do vírus COVID-19, essa validação qualitativa tornou-se inviável. Contudo, a aprendizagem adquirida durante a realização dos experimentos, assim como, a investigação, estudo e entendimento dos dados, possibilitou afirmar a respeito do ambiente de *hardware* e *software* utilizado:

1. A construção do circuito, aguça a criatividade e curiosidade na medida em que é necessário refletir sobre a melhor forma de montagem conforme equipamentos e condições disponíveis;
2. A facilidade na captura dos dados enviados em intervalos de segundos, viabiliza um montante confiável de informação sem a necessidade de confirmação ou repetição de etapas;
3. A configuração automática de um limite para a realização do experimento, traz segurança e proteção tanto para os equipamentos quanto para as pessoas envolvidas no estudo;
4. A plotagem em tempo real de um gráfico, resulta na observação imediata dos efeitos causados pela variáveis de entrada do experimento.

5. A gravação de todos os dados, e seus respectivos instantes de tempo possibilita não só uma análise posterior como a autonomia de todo o processo.

Portanto, com base nas afirmações anteriores há indícios de que o ambiente aberto de *hardware* e *software* construído e utilizado neste projeto é de fato significativo para um aprendizado didático, como também um facilitador de processos experimentais.

6 CONCLUSÃO E TRABALHOS FUTUROS

Estudos sobre a inserção de novas tecnologias no ensino da Física são realizados sobretudo com a intenção de tornar, especialmente para essa disciplina, o ambiente escolar e universitário mais significativo e atrativo para o discente. Nesse aspecto, a plataforma Arduino associada com diferentes tipos de componentes eletrônicos aparece como uma tecnologia versátil, de baixo custo e simples utilização por professores e alunos.

O presente trabalho foi realizado com o objetivo de desenvolver e validar um ambiente significativo aberto de *hardware* e *software*, baseado na utilização do Arduino, para um experimento didático de física de dissipação de calor. Os procedimentos para a construção desse ambiente seguiram as etapas de planejamento do experimento, implementação do *software* e análise dos resultados, descritas detalhadamente no capítulo 5. Em todas as etapas existiram desafios a serem superados, e o estímulo a novos aprendizados também esteve presente durante todo o desenvolvimento do ambiente.

Com base, na construção do circuito, na utilização do sistema e na análise dos dados foi possível verificar as facilidades que um experimento didático automatizado pode proporcionar. Além da velocidade e confiabilidade na aquisição dos dados, o aprendizado crítico e investigativo é incentivado durante todo o processo.

A análise de dados revelou informações sobre a dissipação de calor no transistor TIP31C que teriam uma obtenção mais custosa caso o experimento fosse realizado manualmente. Foi possível verificar e confirmar a grande influência e importância da circulação de ar em um circuito eletrônico e o quanto a falta dela afeta o aquecimento dos componentes, em especial o aquecimento do transistor. Ademais, foi possível identificar durante a execução dos experimentos alguns erros, como também melhorias que poderiam ser futuramente aplicadas, entretanto a facilidade de aprendizado proporcionada pelo próprio ambiente, fez com que parte desses consertos e modificações fossem superados ainda neste projeto.

Assim, concluiu-se que o objetivo deste trabalho foi alcançado na medida em que o ambiente de *hardware* e *software* construído apresentou eficiência do ponto de vista didático, fornecendo informações sólidas sobre o experimento em questão, e causando reflexões interessantes

a respeito da dissipação de calor no transistor TIP31C.

Durante a execução do trabalho, foram analisados os aspectos técnicos de construção do aparato físico, assim como, de implementação do *software*. Alguns pontos de modificação foram identificados com o intuito de proporcionar um melhor controle, manuseio e agilidade no gerenciamento dos experimentos. Como citado anteriormente, algumas dessas melhorias foram possíveis de ser implementadas. Entretanto, outras ficam como sugestão para trabalhos futuros:

1. Construir aparato físico que obtenha maior controle sobre os parâmetros do experimento relacionados a temperatura ambiente e umidade do ar;
2. Modificar a estrutura física do circuito, adicionando isolamento entre os componentes e melhor arquitetura para a coleta de medições elétricas;
3. Aprimorar a administração do experimento, através da criação de novas rotinas do SG, que permitam a pausa e reinicialização do processo, assim como, a execução de vários ensaios em sequência;
4. Aprimorar os arquivos gerados pelo SG, com o intuito de automatizar cálculos estatísticos mais complexos sobre os dados;
5. Permitir maior independência física do ambiente, implementando conexão sem fio e acesso remoto;
6. Aprimorar o SG para torná-lo um software menos específico e capaz de controlar mais tipos de experimentos.

REFERÊNCIAS

ALFACOMPBRASIL. *Alfacombrasil DS18B20 – Conheça o sensor de temperatura digital inteligente*. 2020. Disponível em: <https://alfacompbrasil.com/2019/04/02/ds18b20-conheca-o-sensor-de-temperatura-digital-inteligente/>. Acesso em: 02 março 2020.

ALLDATASHEETS. *DS18B20 Datasheet (PDF) - Dallas Semiconductor*. 2021. Disponível em: <https://www.alldatasheet.com/datasheet-pdf/pdf/205492/DIOTEC/TIP31C.html>. Acesso em: 10 janeiro 2021.

AMORIM, H. S. d.; DIAS, M. A.; SOARES, V. Sensores digitais de temperatura com tecnologia one-wire: Um exemplo de aplicação didática na área de condução térmica. *Revista Brasileira de Ensino de Física*, SciELO Brasil, v. 37, n. 4, p. 4310–1, 2015.

ARADI, P. Offline and online thermostat experiment with labview and arduino. In: IEEE. *2016 International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS)*. [S.l.], 2016. p. 127–131.

ARAUJO, I. S.; VEIT, E. A. Uma revisão da literatura sobre estudos relativos a tecnologias computacionais no ensino de física. *Revista Brasileira de Pesquisa em Educação em Ciências*, v. 4, n. 3, 2004.

ARDUINO. *ARDUINO | ARDUINO UNO REV3*. 2020. Disponível em: <https://store.arduino.cc/usa/arduino-uno-rev3>. Acesso em: 02 março 2020.

ARDUINO. *ARDUINO |What is Arduino?* 2020. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 02 março 2020.

ASHA, K. et al. Real time speed control of a dc motor by temperature variation using labview and arduino. In: IEEE. *2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT)*. [S.l.], 2017. p. 72–75.

BAÚ da Eletrônica. *Plataforma Arduino, universo de possibilidades*. 2020. Disponível em: <http://blog.baudaeletronica.com.br/plataforma-arduino/>. Acesso em: 02 março 2020.

BEZERRA, A. G. et al. Tecnologias livres e ensino de física: Uma experiência na utfpr. In: *XVIII Simpósio Nacional de Ensino de Física – SNEF*. Vitória, ES: [s.n.], 2009.

BRAGA, N. C. *ELETRÔNICA ANALÓGICA*. 1. ed. [S.l.]: Editora Newton C. Braga, 2012. v. 2.

BRAGA, N. C. *ELETRÔNICA BÁSICA*. 1. ed. [S.l.]: Editora Newton C. Braga, 2012. v. 7.

BRAGA, N. C. *SEMICONDUTORES DE POTÊNCIA*. 1. ed. [S.l.]: Editora Newton C. Braga, 2014. v. 7.

BUKSMAN, E. et al. Experimentando con arduino y scilab: propagación de calor en una barra metálica. *Revista Brasileira de Ensino de Física*, SciELO Brasil, v. 41, n. 4, 2019.

CAVALCANTE, M. A.; TAVOLARO, C. R. C.; MOLISANI, E. Física com Arduino para iniciantes. *Revista Brasileira de Ensino de Física*, scielo, v. 33, n. 4, p. 4503, 12 2011. ISSN 1806-1117.

DESAI, P. *Python programming for Arduino*. [S.l.]: Birmingham: Packt Publishing Ltd, 2015.

ELECTRICAL Engineering. *How to fix Heating issue of 7805?* 2020. Disponível em: <https://electronics.stackexchange.com/questions/345566/how-to-fix-heating-issue-of-7805>. Acesso em: 02 março 2020.

FERNÁNDEZ-PACHECO, A.; MARTIN, S.; CASTRO, M. Implementation of an arduino remote laboratory with raspberry pi. In: IEEE. *2019 IEEE Global Engineering Education Conference (EDUCON)*. [S.l.], 2019. p. 1415–1418.

FILIPI Flop. *Sensor de Temperatura DS18B20 a Prova D'água*. 2020. Disponível em: <https://www.filipeflop.com/produto/sensor-de-temperatura-ds18b20-a-prova-dagua/>. Acesso em: 02 março 2020.

GRASSELLI; GARDELLINI. O ensino da Física pela experimentação no ensino médio: da teoria à prática. *Os Desafios da Escola Pública Paranaense na Perspectiva do Professor PDE*, v. 1, 2016.

HECKLER et al. Uso de simuladores, imagens e animações como ferramentas auxiliares no ensino/aprendizagem de óptica. *Revista Brasileira de Ensino de Física*, Scielo, v. 29, p. 267–273, 2007.

HILBERER, A. et al. Temperature-dependent transport measurements with arduino. *Papers in Physics*, v. 10, p. 100007, 2018.

KOESTOER, R. et al. A simple method for calibration of temperature sensor ds18b20 waterproof in oil bath based on arduino data acquisition system. In: . Bali, Indonesia: [s.n.], 2019. v. 2062, p. 020006.

LARA, A. L. de et al. Ensino de física mediado por tecnologias de informação e comunicação: Um relato de experiência. In: *XX Simpósio Nacional de Ensino de Física – SNEF*. São Paulo, SP: [s.n.], 2013.

LESTEIRO-TEJEDA, J.; HERNÁNDEZ-DELFÍN, D.; BATISTA-LEYVA, A. Automation of experiments using arduino. *Revista Cubana de Física*, v. 34, n. 2, p. 120–124, 2017.

LLAMAS, C. et al. Open-source sensors system for doing simple physics experiments. *Papers in physics*, Papers in Physics, v. 10, p. 100004–100007, 2018.

MALVINO, A.; BATES, D. *Eletrônica Volume I*. 8. ed. [S.l.]: AMGH Editora Ltda., 2016. v. 1.

MARGOLIS, M. *Arduino cookbook: recipes to begin, expand, and enhance your projects*. [S.l.]: "O'Reilly Media, Inc.", 2011.

MARTINAZZO, C. A. et al. ARDUINO: UMA TECNOLOGIA NO ENSINO DE FÍSICA. *Perspectiva, Erechim*, v. 38, n. 143, p. 21–30, 2014.

MCROBERTS, M. *Arduino Básico-2^a edição: Tudo sobre o popular microcontrolador Arduino.* [S.l.]: Novatec Editora, 2015.

MOREIRA, I. de C.; PAIVA, M. C. O rei está nu - a palestra de feynman no brasil sobre o ensino de ciências na descrição de oswaldo frota-pessoa. *Física na Escola*, v. 14, n. 1, p. 62, 2016.

MOREIRA, M. et al. Contribuições do arduíno no ensino de física: uma revisão sistemática de publicações na área do ensino. *Caderno Brasileiro de Ensino de Física*, v. 35, n. 3, p. 721–745, 2018.

MOREIRA, M. A. Uma análise crítica do ensino de física. *Estudos Avançados*, scielo, v. 32, n. 94, p. 73 – 80, 12 2018.

NOGUEIRA. *Como utilizar o optoacoplador 4N25 com Arduino*. 2020. Disponível em: <https://autocorerobotica.blog.br/como-utilizar-o-optoacoplador-4n25-com-arduino/>. Acesso em: 02 março 2020.

ORGANTINI, G. Arduino as a tool for physics experiments. In: *Journal of Physics: Conference Series*. [S.l.: s.n.], 2018. v. 1076, n. 1.

PETRY, C. A. et al. Project teaching beyond physics: Integrating arduino to the laboratory. In: IEEE. *2016 Technologies Applied to Electronics Teaching (TAEE)*. [S.l.], 2016. p. 1–6.

PRODANOV, C. C.; FREITAS, E. C. de. *Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição*. [S.l.]: Editora Feevale, 2013.

PYTHON. *Welcome to Python*. 2021. Disponível em: <https://www.python.org/>. Acesso em: 10 janeiro 2021.

RAMALHO; NICOLAU; TOLEDO. *Fundamentos da Física*. 1. ed. [S.l.]: Editora Moderna, 2015. v. 1.

ROSA, C. T. W. D.; TRENTIN, M. A. S.; BIAZUS, M. de O. Tecnologias educacionais no ensino de física: Retrato das pesquisas nacionais. *Revista ENCITEC*, v. 7, n. 2, p. 24–42, 2017.

SABER Atualizado. *Por que o calor vai do quente para o frio?* 2020. Disponível em: <https://www.saberatualizado.com.br/2016/09/por-que-o-calor-vai-do-quente-para-o.html>. Acesso em: 02 março 2020.

SARIK, J.; KYMISSIS, I. Lab kits using the arduino prototyping platform. In: IEEE. *2010 IEEE Frontiers in Education Conference (FIE)*. [S.l.], 2010. p. T3C–1.

SCHULER, C. *Eletrônica I*. 7. ed. [S.l.]: AMGH Editora Ltda., 2013.

SILVEIRA, S.; GIRARDI, M. Development of an experimental Arduino kit for teaching modern physics in high school [Desenvolvimento de um kit experimental com Arduino para o ensino de Física Moderna no Ensino Médio]. *Revista Brasileira de Ensino de Fisica*, v. 39, n. 4, p. e4502, 2017. ISSN 01024744. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85020757911&doi=10.1590%2f1806-9126-RBEF-2016-0287&partnerID=40&md5=6780ed4e6c477dff65ceb832b4d41e55>.

SOUZA, A. R. de et al. The arduino board: a low cost option for physics experiments assisted by pc. *Revista Brasileira de Ensino de física*, SOC BRASILEIRA FISICA CAIXA POSTAL 66328, 05315-970 SAO PAULO, BRAZIL, v. 33, n. 1, 2011.

TECDICAS. *Criando portas lógicas com transistores*. 2020. Disponível em: <https://tecdicas.com/criando-portas-logicas-com-transistores/>. Acesso em: 02 março 2020.

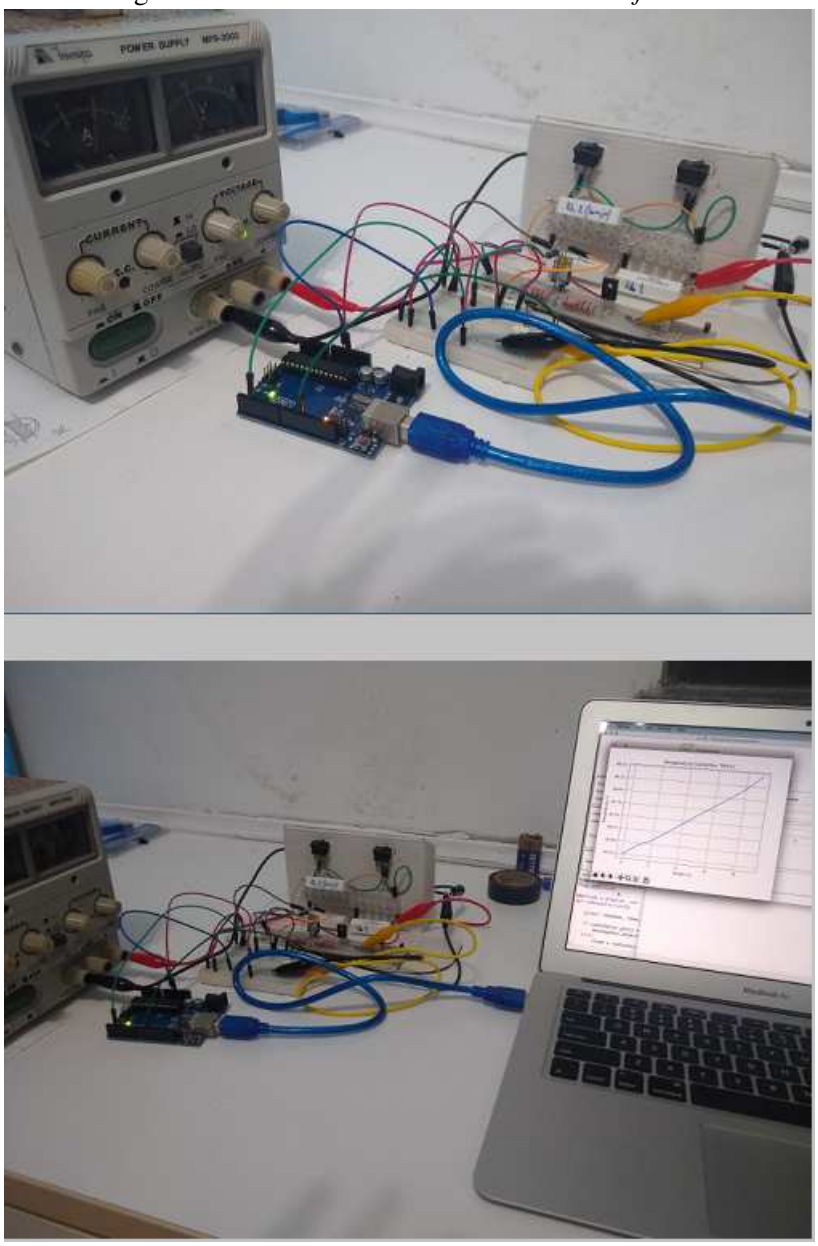
VILAR, A. B. et al. Medição de temperatura: O saber comum ignorado nas aulas experimentais. *Revista Brasileira de Ensino de Física*, v. 37, p. 2507–1, 06 2015.

YOUNG, H. D.; FREEDMAN, R. A. *Física II Termodinâmica e Ondas*. 1. ed. [S.l.]: Editora Pearson, 2016. v. 1.

APÊNDICES

APÊNDICE A – AMBIENTE *HARDWARE* E *SOFTWARE*

Figura 21 – Fotos - Ambiente *Hardware* e *Software*



Fonte – Autor.

APÊNDICE B – CÓDIGO - SISTEMA DE MONITORAMENTO SM

```
1
2  /*Incluir bibliotecas OneWire e DallasTemperature*/
3  #include <OneWire.h>
4  #include <DallasTemperature.h>
5  /*
6
7  * O fio de dados esta conectado ao pino 2 no Arduino
8  * Instancia oneWire para cominicacao com o sensor ds18b20
9  * Referencia oneWire para biblioteca Dallas Temperature.
10 */
11 const int PIN_SENSOR = 2;
12 OneWire oneWire(PIN_SENSOR);
13 DallasTemperature sensors(&oneWire);
14
15 /* Pino digital 11 onde esta conectado o experimento */
16 int PIN_TIP31 = 11;
17
18 /*
19 * Funcao de configuracao void setup()
20 * Configura pino 11 como saida
21 * Inicia comunicacao serial a uma taxa de 9600
22 * Inicia o(s) sensor(es)
23 */
24
25 void setup(void)
26 {
27   pinMode(PIN_TIP31, OUTPUT);
28   Serial.begin(9600);
29   sensors.begin();
```

```
30 }
31
32 void loop(void)
33 {
34   char byteEntrada;                  //variavel para
        armazenar o byte enviado pelo python
35   if (Serial.available()>0){         //Se houver na porta
        serial numero de bytes maior que zero
36     byteEntrada = Serial.read();     //Ler o byte enviado e
         armazena na variavel
37
38     Serial.println("Valor recebido: ");
39     Serial.println(byteEntrada);
40     Serial.println();
41
42     switch(byteEntrada){
43       case "A":
44         /* duty cycle=63 Equivale a 25% */
45         analogWrite(PIN_TIP31, 63);
46         break;
47       case "B":
48         /* duty cycle=63 Equivale a 50% */
49         analogWrite(PIN_TIP31, 127);
50         break;
51       case "C":
52         /* Envia 5V ao pino 11 */
53         digitalWrite(PIN_TIP31, HIGH);
54         break;
55       case "S":
56         /* Envia 0V no pino 11 */
57         digitalWrite(PIN_TIP31, LOW);
58         break;
```

```
59      }
60    }
61      /* Envia o comando para obter a temperatura
62      * Retorna a temperatura na escala Celsius
63      * Index 0 refere-se ao unico sensor conectado ao
            barramento.
64      * Atualiza o valor a cada 1s.
65      */
66      sensors.requestTemperatures();
67      Serial.println(sensors.getTempCByIndex(0));
68      delay(1000);
69  }
```

APÊNDICE C – CÓDIGO - SISTEMA DE GERENCIAMENTO SG

```
1  import serial
2  import time
3  import csv
4  import matplotlib
5  matplotlib.use("tkAgg")
6  import matplotlib.pyplot as plt
7  import numpy as np
8  import tkinter
9  from tkinter import *
10 import sys
11
12 tempmax = 32   #Temperatura maxima
13 name= time.strftime("%Y-%m-%d_%H%M%S")
14 filename = "EXP_%s.csv"% (name) #Nome do Arquivo
15
16 #Porta onde o Arduino esta conectado e taxa de transmissao
      dos dados
17 ser = serial.Serial("COM4", 9600, timeout=1)
18
19 #tkinter canvas
20 #Definicoes da interface com tkinter, top = janela
      principal, Titulo da Janela, Icon da Janela, funcao
      resizable determina se a janela pode ser maximizada ou
      nao
21
22 top = tkinter.Tk()
23 top.title("SG - Sistema de Gerenciamento")
24 top.iconbitmap("images/icon.ico")
25 top.resizable(0, 0)
```

```
26
27 #Sinalizador para manter janela tkinter ativa
28 flag = tkinter.BooleanVar(top)
29 flag.set(True)
30
31 #Abrindo o arquivo .csv
32 def onBotaoIniciar():
33
34     global tempmax, name, filename
35
36     if radioValue.get() == 0:
37         messagebox.showinfo("Atencao", "Selecione valor
              para PWM(%)")
38     else:
39         vlpwm = radioValue.get()  #Armazena valor do
              radiobutton PWM
40
41         with open(filename, "a", newline="") as arquivo_csv
              :
42             writer = csv.writer(arquivo_csv, delimiter=",",
                  dialect="excel")
43             writer.writerow(["Arduino/Dados Porta Serial"])
44
45             #Escrita dos dados digitados pelo usuario
46             writer = csv.writer(arquivo_csv, delimiter=";")
47             writer.writerow(["Curso: ", text_curso.get()])
48             writer.writerow(["Periodo: ", text_turma.get()
                  ])
49             writer.writerow(["Aluno: ", text_aluno.get()])
50             writer.writerow(["Professor: ", text_professor.
                  get()])
51
```

```
52              writer.writerow(["ID : ", text_IdExp.get()])
53              writer.writerow(["Experimento: ",
                   text_Experimento.get()])
54              writer.writerow(["Tensao(V): ", text_tensao.get
                   ()])
55              writer.writerow(["Resistor(Ohm): ",
                   text_resistor.get()])
56              writer.writerow(["PWM(%): ", vlpwm])
57              writer.writerow(["Descricao: ", text_descricao.
                   get("1.0","end-1c")])
58              writer.writerow([""])
59              writer.writerow(["Data: ", time.strftime("%x")
                   ])
60              writer.writerow(["Hora", "Temperatura C"])
61
62          #Instrucoes enviadas ao Arduino determinam tensao e
                iniciam o TIP31
63          if vlpwm == 25:
64              ser.write(b"A") #"A"= PWM 25%
65          elif vlpwm == 50:
66              ser.write(b"B") #"B"= PWM 50%
67          elif vlpwm == 100:
68              ser.write(b"C") #"C"= PWM 100%
69
70          decod_bytes = 0
71          #Enquanto a temperatura recebida for menor ou igual
                a 80.0:
72          while (decod_bytes <= tempmax):
73              if flag.get():
74                      #Recebendo os dados de temperatura do
                           Arduino via serial
75                      ser_bytes = ser.readline()
```

```
76                  print(ser_bytes)
77                  try:
78                      #Decodificando os bytes de dados (
                            bytes para float)
79                      decod_bytes = float(ser_bytes[0:len
                            (ser_bytes)-2].decode("utf-8"))
80
81                      #Visualizacao temperatura na
                            interface
82                      text_temp.config(text = decod_bytes
                            , font=("Arial", 20), fg = "
                            white", bg="#4682B4")
83
84                      #Atualizacao frequente dos valores
                            da interface
85                      text_temp.update_idletasks()
86
87                      #Data atual
88                      text_data.config(text = time.
                            strftime("%x"), font=("Arial",
                            10))
89
90                      #Hora atual
91                      text_hora.config(text = time.
                            strftime("%H:%M:%S"), font=("
                            Arial", 10))
92                      with open(filename,"a", newline="")
                             as arquivo_csv:
93                          writer = csv.writer(arquivo_csv
                                , delimiter=";")
94                          writer.writerow([time.strftime
                                ("%H:%M:%S"),decod_bytes])
```

```
95                      except:
96                          continue
97                      #Zera todos os bytes existentes na fila
                            de entrada.
98                      ser.flushInput()
99                      #Atualiza interface tkinter
100                     top.update()
101             else:
102                 flag.set(True)
103                 break
104         else:
105             text_temp.config(text = decod_bytes, font=("
                    Arial", 20), fg = "white", bg="#FF6347")
106
107             #Instrucao enviada ao Arduino "S" = sair
                    finaliza o experimento.
108             ser.write(b"S")
109
110             tkinter.messagebox.showerror(title="Atencao",
                    message="Temperatura maxima atingida!!")
111             onBotaoParar()
112
113 #Funcao para plotagem do grafico em tempo real
114 def onBotaoGrafico():
115
116     global tempmax, name, filename
117
118     #Matriz para armazenar os valores recebidos pela serial
            .
119     data = np.array([])
120
121     #Criacao da figura onde o grafico e desenhado
```

```
122     plt.figure(num="Grafico - Temperaturas")
123     #Modo interativo para plotagem na tela
124     plt.ion()
125     #Exibe os eventos ativos
126     plt.show()
127
128     decod_bytes=0
129     while (decod_bytes <= tempmax):
130         try:
131             ser_bytes = ser.readline()
132             try:
133                 decod_bytes = float(ser_bytes[0:len(
                        ser_bytes)-2].decode("utf-8"))
134                 text_temp.config(text = decod_bytes, font
                        =("Arial", 20))
135                 upd_label.update_idletasks()
136                 with open(filename,"a", newline="") as
                        arquivo_csv:
137                     writer = csv.writer(arquivo_csv,
                            delimiter=";")
138                     writer.writerow([time.strftime("%H:%M:%
                            S"),decod_bytes])
139             except:
140                 continue
141
142             plt.title ("Temperatura transistor TIP31C")
143             plt.xlabel("Tempo (s)")
144             plt.ylabel("Temperatura C")
145             plt.grid(True)
146
147             #Acrescenta os valores de temperatura ao final
                    da matriz
```

```
148             data = np.append(data, decod_bytes)
149
150             print(data)
151             #Plota o grafico
152             plt.plot(data, "b-")
153             #Pausa para salvar/redimensionar grafico
154             plt.pause(0.01)
155         except:
156             break
157     else:
158         text_temp.config(text = decod_bytes, font=("Arial",
                20), fg = "white", bg="red")
159         ser.write(b"S")
160         tkinter.messagebox.showerror(title="Atencao",
               message="Temperatura maxima atingida!!")
161         onBotaoParar()
162
163 #Evento do Botao Sair - Fecha o programa e a comunicacao
    serial
164 def onBotaoParar():
165     ser.write(b"S")
166     if messagebox.askokcancel("Sair", "Experimento
           finalizado. Sair?"):
167         print ("Saindo....")
168         flag.set(False)
169         ser.close()
170         top.quit()
171         top.destroy()
172         print ("Fim.")
173         sys.exit()
174
175
```

```
176 #Configuracoes de interface utilizando o tkinter - Labels,
       imagens, frames, caixas de texto e botoes.
177 img = PhotoImage(file="images/logog.png")
178 label_imagem = Label(top, image=img).grid(row=0, sticky=W,
       columnspan=3)  #Label imagem UNEB
179
180 label_titulo = Label(top,
181                      text = "Sistema de Gerenciamento -
                           Experimento com ARDUINO",
182                      font = "Arial 10 bold\n",
183                      bd = 5,
184                      relief="ridge",
185                      width=75,
186                      fg="#4682B4",
187                      padx=5,
188                      pady=5
189 ).grid(row=1, columnspan=3)
190
191 #Frame Informacoes
192 frame_i = LabelFrame(top, text="Informacoes Gerais", padx
       =5, pady=5, borderwidth = 2)
193 frame_i.grid(sticky=EW)
194
195 #Frame Experimento
196 frame_e = LabelFrame(top, text="Informacoes Experimento",
       padx=5, pady=5, borderwidth = 2)
197 frame_e.grid(sticky=EW)
198
199 #Frame Iniciar
200 frame_in = LabelFrame(top, text="Executar", padx=5, pady=5,
        borderwidth = 2)
201 frame_in.grid(sticky=EW)
```

```
202
203 #Frame Botoes
204 frame_bt = LabelFrame(top, padx=5, pady=5, borderwidth = 1)
205 frame_bt.grid(sticky=EW)
206
207 #Labels Informacoes Gerais
208 label_curso = Label(frame_i, text="Curso:", width=5, height
    =1, anchor=W, padx=5, pady=5).grid(row=2, column=0,
    sticky=W)
209 label_turma = Label(frame_i, text="Periodo:", width=6,
    height=1, anchor=W, padx=5, pady=5).grid(row=2, column
    =2)
210
211 label_aluno = Label(frame_i, text="Aluno(a):", width=7,
    height=1, anchor=W, padx=5, pady=5).grid(row=3, column
    =0, sticky=W)
212 label_professor = Label(frame_i, text="Professor(a):",
    width=10, height=1, anchor=W, padx=5, pady=5).grid(row
    =4, column=0, sticky=W)
213
214 #Caixas de Texto - Frame Informacoes Gerais
215 text_curso = Entry(frame_i, width=60)
216 text_curso.grid(row=2, column=1)
217
218 text_turma = Entry(frame_i, width=15)
219 text_turma.grid(row=2, column=3)
220
221 text_aluno = Entry(frame_i, width=85)
222 text_aluno.grid(row=3, column=1, columnspan=3, sticky=W)
223
224 text_professor = Entry(frame_i, width=85)
```

```
225 text_professor.grid(row=4, column=1, columnspan=3, sticky=W
    )
226
227 #Labels Experimento
228 label_IdExp = Label(frame_e, text="Codigo:", width=10,
    height=1, anchor=W, padx=5, pady=5).grid(row=5, column
    =0)
229 label_Experimento = Label(frame_e, text="Experimento:",
    width=10, height=1, anchor=W, padx=5, pady=5).grid(row
    =5, column=2, sticky=W)
230
231 label_tensao = Label(frame_e, text="Tensao(V):", width=10,
    height=1, anchor=W, padx=5, pady=5).grid(row=6, column
    =2)
232 label_resistor = Label(frame_e, text="Resistor(Ohm):",
    height=1, padx=5, pady=5).grid(row=6, column=3)
233 label_pwm = Label(frame_e, text="PWM(%):", width=10, height
    =1, anchor=W, padx=5, pady=5).grid(row=6, column=0)
234 label_descricao = Label(frame_e, text="Descricao:", width
    =10, height=1, anchor=W, padx=5, pady=5).grid(row=7,
    column=0)
235
236 #Caixas de Texto - Frame Informacoes Experimento
237 text_IdExp = Entry(frame_e, width=20)
238 text_IdExp.grid(row=5, column=1, sticky=W)
239
240 text_Experimento = Entry(frame_e, width=50)
241 text_Experimento.grid(row=5, column=3, sticky=W)
242
243 text_tensao = Entry(frame_e, width=15)
244 text_tensao.grid(row=6, column=3, sticky=W)
245
```

```
246 text_resistor = Entry(frame_e, width=15)
247 text_resistor.grid(row=6, column=3, sticky=E)
248
249 #Radiobutton PWM
250 radioValue = IntVar()
251
252 rdiopwn_0 = Radiobutton(frame_e, text="25",
253                         variable=radioValue, value=25)
254 rdiopwn_0.grid(column=1, row=6,sticky=W)
255
256 rdiopwn_1 = Radiobutton(frame_e, text="50",
257                         variable=radioValue, value=50)
258 rdiopwn_1.grid(column=1, row=6)
259
260 rdiopwn_2 = Radiobutton(frame_e, text="100",
261                         variable=radioValue, value
                               =100)
262 rdiopwn_2.grid(column=1, row=6,sticky=E)
263
264 text_descricao = Text(frame_e, width=64, height=5)
265 text_descricao.grid(row=7, column=1, columnspan=3,sticky=W)
266
267 #Labels Executar
268 label_temp = Label(frame_in, text="Temperatura(C):",width
      =13, height=1, anchor=W, padx=5, pady=5).grid(row=8,
      column=0, rowspan=2)
269 label_hora = Label(frame_in, text="Hora:", width=5, height
      =1, anchor=W, padx=5, pady=5).grid(row=8, column=2)
270 label_data = Label(frame_in, text="Data:", width=5, height
      =1, anchor=W, padx=5, pady=5).grid(row=9, column=2)
271
272 #Caixas de Texto - Frame Executar
```

```
273 text_temp = tkinter.Label(frame_in, text=" ", width=10,
       relief="sunken")
274 text_temp.grid(row=8, column=1, sticky=W, rowspan=2)
275
276 text_hora = tkinter.Label(frame_in, text=" ", width=10,
       relief="sunken")
277 text_hora.grid(row=8, column=3)
278
279 text_data = tkinter.Label(frame_in, text=" ", width=10,
       relief="sunken")
280 text_data.grid(row=9, column=3)
281
282 #Botoes
283 botao_Iniciar = tkinter.Button(frame_bt, text="Iniciar",
       command=onBotaoIniciar).grid(row=10)
284 botao_Parar = tkinter.Button(frame_bt, text="Sair", command
       =onBotaoParar).grid(row=10, column=1)
285 botao_Grafico = tkinter.Button(frame_bt, text="Grafico",
       command=onBotaoGrafico).grid(column=2, row=10)
286
287 top.mainloop()
```