

UNIVERSIDADE ESTADUAL PAULISTA
FACULDADE DE ENGENHARIA DE BAURU
DEPARTAMENTO DE ENGENHARIA MECÂNICA

RODRIGO EDUARDO PREDOLIN

**DESENVOLVIMENTO DE UM SISTEMA DE AQUISIÇÃO DE
DADOS USANDO PLATAFORMA ABERTA**

BAURU – SP

2017

RODRIGO EDUARDO PREDOLIN

**DESENVOLVIMENTO DE SISTEMA DE AQUISIÇÃO
DE DADOS USANDO PLATAFORMA ABERTA**

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Mecânica da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de Bauru, como parte dos requisitos para obtenção do título de Mestre em Engenharia Mecânica.

Orientador: Prof. Dr. Vicente Luiz Scalon

BAURU – SP

2017

FICHA CATALOGRÁFICA

Predolin, Rodrigo Eduardo.

Desenvolvimento de um sistema de aquisição de dados usando plataforma aberta/ Rodrigo Eduardo Predolin, 2017

87 f. : il.

Orientador: Vicente Luiz Scalon

Dissertação (Mestrado)-Universidade Estadual Paulista. Faculdade de Engenharia, Bauru, 2017

1. Sensor coletor solar. 2. Anemômetro. 3. Arduino. 4. Baixo custo. 5. Módulo coleta de dados I. Universidade Estadual Paulista. Faculdade de Engenharia. II. Título.

ATA DA DEFESA PÚBLICA DA DISSERTAÇÃO DE MESTRADO DE RODRIGO EDUARDO PREDOLIN, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA, DA FACULDADE DE ENGENHARIA - CÂMPUS DE BAURU.

Aos 14 dias do mês de agosto do ano de 2017, às 09:00 horas, no(a) Anfiteatro da Diretoria Técnica de Informática/FEB, reuniu-se a Comissão Examinadora da Defesa Pública, composta pelos seguintes membros: Prof. Dr. VICENTE LUIZ SCALON - Orientador(a) do(a) Departamento de Engenharia Mecânica / Faculdade de Engenharia de Bauru, Prof. Dr. GERALDO LUIZ PALMA do(a) Departamento de Engenharia Mecânica / Faculdade de Engenharia de Bauru - UNESP, Prof. Dr. PAULO CESAR MIORALLI do(a) Departamento de Indústria / Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP, sob a presidência do primeiro, a fim de proceder a arguição pública da DISSERTAÇÃO DE MESTRADO de RODRIGO EDUARDO PREDOLIN, intitulada **DESENVOLVIMENTO DE UM SISTEMA DE AQUISIÇÃO DE DADOS USANDO PLATAFORMA ABERTA**. Após a exposição, o discente foi arguido oralmente pelos membros da Comissão Examinadora, tendo recebido o conceito final: Aprovado . Nada mais havendo, foi lavrada a presente ata, que após lida e aprovada, foi assinada pelos membros da Comissão Examinadora.

Prof. Dr. VICENTE LUIZ SCALON

Prof. Dr. GERALDO LUIZ PALMA

Prof. Dr. PAULO CESAR MIORALLI

AGRADECIMENTOS

Ao meu orientador Prof. Dr. Vicente Luiz Scalon pelo apoio, confiança, amizade e ensinamentos concedidos e tempo dedicado a este trabalho.

A minha esposa, Danusa Renata Costa Predolin e minha filha, Julia Costa Predolin, pelo apoio e paciência neste período de dedicação ao estudo e a pesquisa.

Aos meus pais, Antonio Carlos Predolin e Marisa Carlini Predolin, que me ensinaram a superar os obstáculos e se dedicar ao estudo, além do carinho fornecido ao longo de toda a minha vida.

Aos meus familiares e amigos que me apoiaram e me incentivaram.

A todos os professores e funcionários da Unesp, em especial a Faculdade de Engenharia de Bauru e a Seção Técnica de Pós-graduação da Faculdade de Engenharia de Bauru, que me auxiliaram no aprendizado adquirido no período de graduação e pós-graduação.

PREDOLIN, R. E. **Desenvolvimento de sistema de aquisição de dados usando plataforma aberta**. 2017. 94 f. Dissertação (Mestrado em Engenharia Mecânica) – Faculdade de Engenharia, Universidade Estadual Paulista, Bauru, 2017.

RESUMO

Sistemas para monitoramento de parâmetros são amplamente utilizados no setor industrial para controle de processos e também na área de pesquisa e desenvolvimento. Estes sistemas permitem uma análise detalhada do comportamento de equipamentos e dispositivos e fornecem informações que auxiliam na melhora do seu desempenho. No estudo da utilização de energias renováveis o uso de sensores é aplicado para mapear o ambiente onde o equipamento está inserido e analisar o seu comportamento e desempenho. No caso de coletores solares, é realizado o monitoramento da temperatura do ambiente, da água no coletor em locais diferentes, da velocidade do vento e da radiação solar, permitindo o seu controle e melhorando o seu desempenho. A instrumentação adequada pode trazer melhorias aos coletores solares, porém dependem de estudos detalhados do seu comportamento através da aquisição de dados do equipamento e do ambiente onde ele está inserido. Para viabilizar esta otimização são necessários equipamentos de coleta de dados específicos que, normalmente, tem alto custo de aquisição. Este é um dos principais empecilhos para uma maior evolução destes equipamentos, principalmente os destinados ao uso residencial. Sendo assim, este trabalho objetiva desenvolver um módulo de coleta de dados de baixo custo para auxiliar no estudo de equipamentos em diversas áreas, incluindo a área de fontes de energia renovável. Um dispositivo deste tipo possibilita a coleta de diversos dados físicos como, por exemplo, a temperatura, a velocidade do vento e vazão d'água. Com o dispositivo há a possibilidade do armazenamento desses dados em um cartão SD, facilitando a sua transferência para o computador. Ao final do trabalho é apresentado o projeto de um módulo e as bibliotecas de maneira a permitir a sua fácil utilização. Alguns valores de parâmetros também foram captados, verificando-se o comportamento adequado do módulo e biblioteca nas condições propostas.

Palavras-chave: sensor coletor solar, anemômetro, Arduino, baixo custo, módulo de coleta de dados.

PREDOLIN, R. E. **Development of data acquisition system using open platform**. 2017. 94 f. Dissertação (Mestrado em Engenharia Mecânica) – Faculdade de Engenharia, Universidade Estadual Paulista, Bauru, 2017.

ABSTRACT

Parameter monitoring systems are widely used in the industrial sector for process control and also in the area of research and development. These systems afford device's behavior analysis with detailed and provide information that helps improve its performance. In the study of renewable energies use, the sensors is applied to map the environment where the equipment is inserted and to analyze its behavior and performance. In the case of solar collectors, the monitoring occurs in the environment temperature, collector water temperature in different places, wind speed and the solar radiation, allowing its control and improving its performance. Proper instrumentation can bring improvements to solar collectors, but depend on detail studies of its behavior based on the acquisition of data from the equipment and the environment where it is inserted. To make this optimization feasible, specific data collection equipment is required, which normally has a high acquisition cost. This is one of the main impediments to keep the evolution of this equipment, especially those destined for residential use. Therefore, this work aims to develop a low cost data collection module to aid in the initial study of equipments used in several sector, include the renewable energy sources sector. This kind of device turn possible the collection of various physical data, such as temperature, wind speed and water flow. With this device there is the possibility of storing this data on an SD card, making it easy to transfer to the computer. At the end of the work, the design of the module and its libraries are presented in a way that allows easy use. Some parameter values were also captured, verifying the proper behavior of the module and library in the proposed conditions.

Keywords: solar sensor collector, anemometer, Arduino, low cost, data collection module.

Sumário

RESUMO.....	2
ABSTRACT.....	3
LISTA DE FIGURAS.....	6
LISTA DE TABELAS.....	8
I - Introdução.....	10
I.1. Aplicações em energias renováveis.....	12
II - Revisão Bibliográfica.....	16
III - Materiais e Métodos.....	22
III.1. Sensores Hall.....	23
III.2. Sensor de Vazão.....	23
III.2.1. Programação sensor de vazão.....	25
III.3. Anemômetro.....	29
III.3.1. Programação anemômetro.....	33
III.4. Direção do Vento.....	36
III.4.1. Programação do sensor direção do vento.....	39
III.5. Sensor Termopar.....	41
III.5.1. Programação sensor termopar para medidas de diferenças de temperaturas.....	45
III.6. Módulo de Calendário e Hora.....	47
III.7. Sensor de Temperatura DS18B20.....	49
IV - Resultados.....	55
IV.1. Elaboração de Bibliotecas.....	55
IV.2. Calibração do anemômetro de caneca.....	59
IV.3. Montagem do módulo de coleta de dados.....	61

IV.3.1. Armazenamento dos dados.....	67
V - Conclusões e trabalhos futuros.....	69
Referências Bibliográficas.....	72
Apêndice A – Bibliotecas do sistema.....	67
Apêndice B – Exemplo de programação para uso do módulo de coleta de dados.....	76
Apêndice C – Representação elétrica do módulo de coleta de dados.....	82

LISTA DE FIGURAS

Figura 1: Esboço de como a saída de energia de uma turbina eólica varia com a velocidade constante do vento.....	13
Figura 2: Coletor solar com monitoramento (á esquerda) e interface de monitoramento (á direita).	14
Figura 3: Esquema de um termopar coaxial. Adaptado de Desikan et al. , 2016.....	18
Figura 4: Análise do sinal gerado pelo sensor de vazão.....	24
Figura 5: Módulo de processamento de sinal, Arduino, com o resistor Pull Up e sensor de vazão durante os testes da programação.....	25
Figura 6: Esquema elétrico para acoplar o sensor de vazão ao processador de sinal.....	25
Figura 7: Fluxograma da programação sensor de vazão.....	27
Figura 8: Fluxograma da função programação com a contagem do tempo entre os pulsos.....	29
Figura 9: Dimensões das canecas do anemômetro.....	31
Figura 10: Anemômetro de 3 canecas com sensor tipo Hall e três acionadores do sensor por volta.	31
Figura 11: Sensor de efeito Hall inserido no alojamento e desenho do alojamento na fase de projeto.....	32
Figura 12: Alojamento dos ímãs e rolamento – fase de projeto.....	32
Figura 13: Esquema elétrico para acoplar o anemômetro com sensor Reed Switch ao processador de sinal.....	33
Figura 14: Fluxograma da programação do anemômetro de 3 canecas com 3 pulsos por volta.....	36
Figura 15: Desenho do leme com destaque da sua estrutura interna de honeycomb.....	37
Figura 16: Encoder multivoltas.....	38
Figura 17: Alojamento do encoder, situado na base do equipamento.....	38
Figura 18: Sistema para medição da velocidade e posição do vento.....	39
Figura 19: Fluxograma da programação do sensor direção do vento.....	41
Figura 20: circuito utilizado por Seebeck.....	41
Figura 21: curvas de temperatura x milivoltagem.....	42

Figura 22: montagem de termopares com 3 junções para medição da temperatura T com base na temperatura de referência Tref. (Holman, J. P.,2012).....	43
Figura 23: Croqui da ligação elétrica do módulo ADC ao arduino UNO.....	44
Figura 24: Teste do conversor analógico digital.....	46
Figura 25: CI MAX6675 a esquerda na imagem e sua montagem em uma placa de fenolite com os bornes de conexão e capacitor de 0,1µF.....	47
Figura 26: Módulo relógio e calendário DS1307.....	48
Figura 27: Ligação de múltiplos sensores de temperatura DS18B20.....	50
Figura 28: Lógica de operação da programação que identifica o endereço do sensor Dallas DS18B20.....	52
Figura 29: Curvas de calibração do anemômetro de caneca com linha de tendencia com equações linear, de segundo e terceiro grau e exponencial.....	62
Figura 30: Módulo de coleta de dados.....	64
Figura 31: módulo de coleta de dados em funcionamento.....	66
Figura 32: dados coletados e armazenados no cartão SD.....	69
Figura 33: Gráfico com os dados coletados e exportado para LibreOfficeCalc.....	70
Figura 34: Ligação elétrica dos componentes do módulo representado na placa protoboard.....	94
Figura 35: Placa de circuito impresso do módulo de coleta de dados: trilha de cobre e identificação dos componentes.....	95
Figura 36: Placa de circuito impresso do módulo de coleta de dados: trilhas de conexão dos componentes.....	96
Figura 37: Placa de circuito impresso do módulo de coleta de dados: identificação dos componentes para impressão.....	97

LISTA DE TABELAS

Tabela 1: Sensores mais usados e suas principais vantagens e desvantagens.....	12
Tabela 2: Coeficientes polinomiais para a Eq. 4 para várias combinações de termopares padrão.....	43
Tabela 3: Valores estatísticos para amostragem com diferentes equações da linha de tendencia.....	61
Tabela 4: Identificação da ligação dos sensores no Arduino UNO no módulo de coleta de dados....	66
Tabela 5: Dados coletados pelo módulo e armazenado no cartão SD após importados pelo software LibreOfficeCalc.....	68

SIMBOLOGIA

Letras Latinas

A	Coeficiente de calibração da equação de 1º grau do anemômetro $\left[\frac{km/h}{Hz} \right]$
ADC	Conversor analógico digital
B	Coeficiente de calibração da equação de 1º grau do anemômetro [km/h]
C_i	Coeficientes polinomiais para combinações de termopares
CI	Circuito integrado
CS	Conector dedicado a seleccionar o periférico que se comunicará com a placa Arduino usando o protocolo de comunicação SPI
CSV	Comma-separated values: formato de arquivo que armazena dados tabelados
d	Numero de amostras
DQ	Conector para transmissão de dados digitais em protocolo de comunicação ONEWIRE
E	Diferença de potencial proporcional a temperatura [V]
f	Frequência de pulsos do anemômetro [Hz]
FEM	Força eletro motriz [V]
f_r	Frequência de pulsos totais do anemômetro [Hz]
I2C	Protocolo de comunicação entre a placa Arduino e os periféricos
MISO	Conector dedicado a receber dados dos periféricos para a placa Arduino que usam protocolo de comunicação SPI
MOSI	Conector dos periférios dedicado a receber dados da placa Arduino que usam protocolo de comunicação SPI
m_s	Média da amostragem
n	Número da amostragem
N_p	Numero de pulsos por rotação do anemômetro
PLA	Poliácido láctico ou ácido poliláctico: polímero composto de ácido láctico
PET	Polietileno tereftalato: polímero termoplástico
S_A	Coeficiente de Seebeck do metal A do termopar
S_B	Coeficiente de Seebeck do metal B do termopar
SCK	Conector que sincroniza o clock da placa Arduino com os periféricos que usam protocolo de comunicação SPI
SCL	Conector que sincroniza o clock da placa Arduino com os periféricos que usam protocolo de comunicação I2C

SD	Cartão de memória Secure Digital Card
SDA	Conector de dados em periféricos que usam protocolo de comunicação I2C
SPI	Protocolo de comunicação entre a placa Arduino e os periféricos
SS	Conector dedicado a selecionar o periférico que se comunicará com a placa Arduino usando o protocolo de comunicação SPI
T	Temperatura [°C]
T1	Temperatura na junção 1 do termopar [°C]
T2	Temperatura na junção 2 do termopar [°C]
V	Velocidade do vento [km/h]
V ⁱ	Voltagem termoelétrica com a junção de referencia [V]
x _i	Valor da amostra i

Letras Gregas

σ_e Desvio padrão

I - INTRODUÇÃO

O termo instrumentação é normalmente utilizado para se referir a dispositivos e equipamentos utilizados para medir, registrar, converter e transmitir variáveis de um fenômeno, para posteriormente serem avaliados e utilizados como parâmetros no controle. As principais grandezas físicas mensuráveis e de maior interesse neste trabalho são a vazão, a pressão, a força, a temperatura, a velocidade e a umidade.

A implementação de um dispositivo de medição, controle e monitoramento pode ser complexa, pois precisa garantir a precisão na leitura, confiabilidade e repetibilidade. Esses dispositivos são utilizados em diversos dispositivos como, por exemplo, aquecedores, reatores, bombas, prensas, fornos, refrigeradores, condicionadores de ar e em muitos outros equipamentos. Seu objetivo é sempre a obtenção de grandezas físicas ou químicas por meio de unidades de medidas apropriadas para detectar e registrar essas informações. .

A leitura da grandeza física é realizada por dispositivos identificados como sensores, que as transformam em variáveis convenientes, que em muitos casos são elétricas, associando a grandeza física a uma tensão e/ou corrente. Há sensores em que a amplitude do sinal elétrico de saída reproduz a amplitude do sinal de entrada, esses dispositivos são identificados como sensores analógicos (MORAIS e CASTRUCCI, 2007). A saída elétrica destes sensores pode ser ainda na forma digital, realizada através de uma codificação binária que precisa, posteriormente, ser interpretada.

Os sensores digitais podem ainda ser de dois tipos distintos, o de contato físico e sem o contato físico, denominados como sensores de proximidade, os quais são de grande relevância e podem ser aplicados em diversas situações pelo seu menor desgaste com o uso e, portanto, com uma maior confiabilidade.

Para os sensores de proximidade existe 5 princípios de funcionamento, sendo eles:

- Indutivo: usam correntes induzidas por campo magnético com o objetivo de detectar objetos metálicos por perto. Eles usam uma bobina para gerar um campo magnético de alta frequência;
- Capacitivo: detecta alterações em um campo eletrostático e é apropriado para detectar objetos não metálicos e metálicos. São semelhantes aos sensores indutivos, com a diferença de produzir um campo eletrostático em lugar de um campo eletromagnético;
- Ultrassônico: usa ondas acústicas e ecos, apropriados para objetos grandes;
- Fotoelétrico: detecta variações de luz infravermelha recebida;
- Efeito Hall: detecta alterações de campo magnético. Constituídos de geradores de tensão Hall, amplificadores de sinal, circuito de disparo e circuitos de saída com transistor num só circuito integrado e podem operar em frequências de até 100 kHz, tendo custo muito menor que as chaves eletromecânicas.

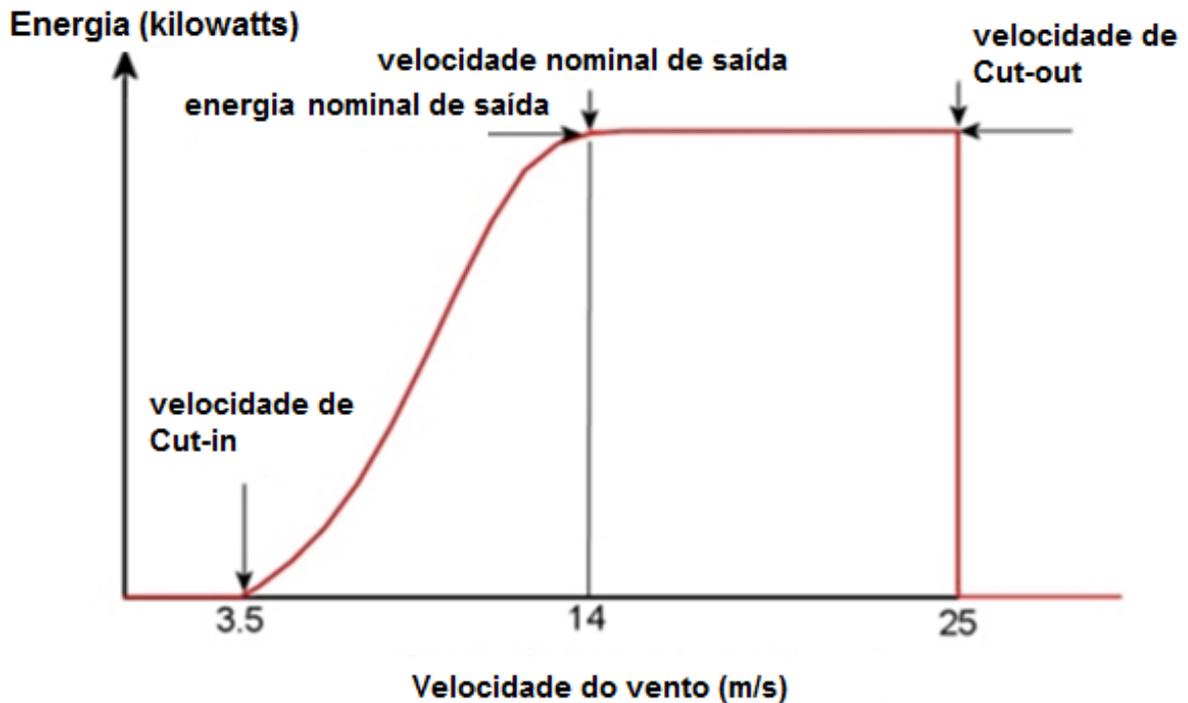
A tabela 1 mostra os principais sensores e as suas vantagens e desvantagens.

Tabela 1: Sensores mais usados e suas principais vantagens e desvantagens.
 Fonte: MORAIS e CASTRUCCI, 2007

Classificação	Sensores	Vantagens	Desvantagens
Sensores de Proximidade	Sensores de contato Chaves de contato	<ul style="list-style-type: none"> - Capacidade de corrente - Imunidade à interferência - Baixo custo - Tecnologia conhecida 	<ul style="list-style-type: none"> - Requer contato físico com o alvo - Resposta lenta - Contato apresentam vida curta - Movimento produz desgaste
	Indutivos	<ul style="list-style-type: none"> - Resiste a ambientes severos - Muito previsível - Vida longa - Fácil instalação - Não depende da superfície do objeto 	<ul style="list-style-type: none"> - Limitação de distância - Detecta principalmente materiais metálicos - Sensível a interferência eletromagnética
	Capacitivos	<ul style="list-style-type: none"> - Detecção através de algumas embalagens - Pode detectar materiais não metálicos - Vida longa 	<ul style="list-style-type: none"> - Distância curta de detecção - Muito sensível a mudanças ambientais - Não é seletivo em relação ao alvo
	Óticos	<ul style="list-style-type: none"> - Pode ser usado com qualquer material - Vida longa - Faixa grande de medição - Resposta rápida - Pode retirar o ruído ambiental - Permite o uso de fibras óticas 	<ul style="list-style-type: none"> - Lentes sujeitas à contaminação - Faixa afetada pela cor e refletividade do alvo - Mudança de ponto focal pode modificar o desempenho - Objetos brilhantes podem interferir
	Ultra-sônicos	<ul style="list-style-type: none"> - Pode medir distâncias longas - Pode ser usado para detectar muitos materiais - Resposta linear com a distância 	<ul style="list-style-type: none"> - Requerem um alvo com área mínima de trabalho - Apresentam distâncias mínimas de trabalho - Resolução depende da frequência - Sensível a mudanças do ambiente - Não funciona com materiais de densidade
	Hall	<ul style="list-style-type: none"> - Vida longa - Fácil instalação - Resposta rápida - Baixo custo 	<ul style="list-style-type: none"> - Não é seletivo em relação ao alvo - O alvo deve ter um ímã fixado - Sensível a interferências eletromagnéticas

I.1. Aplicações em energias renováveis

No setor de energia renováveis, os sensores são fundamentais para analisar o desempenho dos equipamentos e as condições climáticas que atua sobre eles. Por exemplo, a coleta de dados que irá definir a viabilidade econômica para a instalação de geradores eólicos e após a sua instalação é usada no levantamento das condições climáticas. Durante a operação do sistema eólico de geração de energia, a determinação da velocidade do vento é usada para liberação do funcionamento da turbina, ou seja, quando esta for suficiente para iniciar o movimento de giro da hélice (cut-in). O oposto também é usado, isto é, quando a velocidade do vento for muito alta, a turbina eólica interrompe o seu funcionamento (cut-out), conforme representado na figura 1, de forma a manter a integridade dos seus componentes mecânicos. Em equipamentos mais complexos, a velocidade do vento é usada não somente para iniciar e parar o giro das pás, mas também para otimizar o funcionamento da turbina, alterando a sua geometria e velocidade.



Adaptado de: http://www.wind-power-program.com/turbine_characteristics.htm

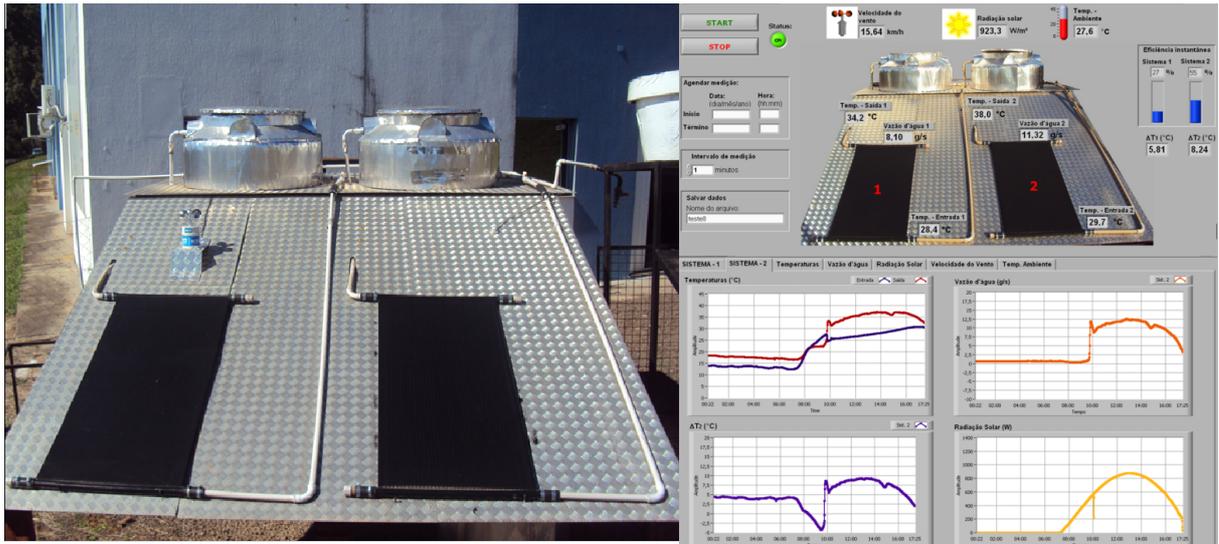
Figura 1: Esboço de como a saída de energia de uma turbina eólica varia com a velocidade constante do vento.

Sistema semelhante de geração de energia também pode ser utilizado em residências. Estes sistemas são identificados como sistema eólico de pequeno porte, embora sejam mais conhecidos como micro e minigeradores eólicos. Segundo a resolução normativa 482/2012 da ANEEL, microgeradores são definidos como sistemas com potência de até 75 kW e minigeradores, acima de 75 kW até 3 MW, conectados à rede de distribuição por meio de instalações de unidades consumidoras.

Antes de instalar um sistema de geração de energia de pequeno porte de fontes renováveis, será preciso fazer um estudo econômico para ter certeza qual sistema de geração de energia é viável. Porém, para isso ocorrer, é necessário um estudo das condições climáticas do local onde o equipamento será instalado, ou seja, nas próprias residências. Entretanto, esse estudo dificilmente é realizado devido à dificuldade de se obter equipamentos específicos na coleta de informações sobre as condições ambientais, tais como velocidade do vento, temperatura e radiação solar.

Outro sistema amplamente utilizado de energia renovável é o aquecimento solar de água, composto basicamente por um coletor solar, tanque de armazenamento de água quente

devidamente isolado e fonte de água fria para reposição. Um aumento do desempenho destes sistemas pode ser obtido usando controle e monitoramento através de instrumentos como termômetros, radiômetros, anemômetros instalados em um módulo de controle, conforme apresentado na figura 2. Apesar de efetivo, este tipo de conjunto é raramente aplicado devido à elevação que provoca no custo do sistema.



Fonte: <http://sine.ni.com/cs/app/doc/p/id/cs-16174#>

Figura 2: Coletor solar com monitoramento (à esquerda) e interface de monitoramento (à direita)

Aplicações práticas dos sensores descritos não se restringem ao setor industrial, na área de pesquisa e desenvolvimento eles são largamente utilizados. Com o seu auxílio é possível compreender e analisar o comportamento de diversos equipamentos e dispositivos, fornecendo informações relevantes que auxiliem no melhoramento de seu desempenho.

Os coletores solares de aquecimento de água residencial no Brasil foram analisados pelo Instituto Nacional de Metrologia, Normalização e Qualidade industrial (INMETRO), conforme programa brasileiro de etiquetagem e publicação das *tabelas de consumo/eficiência energética, edição 03/16*, mediante regulamento e metodologia específica para uso da etiqueta nacional de conservação de energia, 283 coletores solares comerciais destinados ao aquecimento de água para o banho foram analisados, demonstrando que a eficiência energética média foi no máximo de 67,5%, variando este valor para o mínimo de 34,3%, dependendo do modelo analisado. Esses dados demonstram que melhorias nos coletores solares são possíveis, porem dependem de estudos detalhados do seu comportamento com sistemas de aquisição de dados do equipamento e do ambiente onde ele está inserido.

Assim, com o cenário favorável ao estudo de fontes renováveis de energia e o seu aprimoramento e monitoramento, o presente trabalho objetiva o desenvolvimento de um dispositivo de coleta e armazenamento de dados e uma biblioteca de códigos de programação. Este desenvolvimento pode ser aplicado no aprimoramento de diversos dispositivos, incluindo o sistema inteligente de monitoramento e controle de operação de coletores solares.

II - REVISÃO BIBLIOGRÁFICA

Sensores analógicos e digitais são usados na medição de quantidades físicas e elétricas, dentre elas, a temperatura, pressão, massa e tensão. Além da medida, dispositivos podem ainda serem aplicados para monitorar e controlar determinada variável. O primeiro elemento de um sistema de medição é o sensor de entrada, um dispositivo que permite a conversão de uma “energia” de uma forma para outra mais adequada. Uma definição mais ampla pode substituir "informação" por "energia".(NORTHROB, 2005,p.215)

Conforme citado por Thomazini e Albuquerque (2011), o estudo do sensoriamento e controle industrial, comercial e automobilístico, permite a obtenção de variáveis físicas do ambiente monitorado. Alguns sensores utilizados para este fim não possuem as características necessárias para utilização direta em um sistema de controle. A causa principal desta impossibilidade se deve ao fato do sinal produzido ter um valor muito baixo e para a sua utilização, é necessário a amplificação do sinal antes de sua análise.

Conforme Northrop (2005), praticamente todos os sistemas de instrumentação requerem algum tipo de condicionamento do sinal analógico entre a entrada do sinal no transdutor e o sistema de processamento e armazenamento. A forma mais simples de condicionamento do sinal e que atende a maior parte dos casos é a amplificação do sinal de tensão através de amplificadores operacionais, o que provoca uma alteração no nível de impedância entre a entrada e saída.

Segundo Alciatore e Histan (2012), um circuito típico de um amplificador operacional geralmente inclui *feedback* da saída do sinal para a entrada negativa (amplificador operacional inverso). Este tipo de configuração é chamada de loop fechado e resulta na estabilidade do amplificador e controle do ganho. Quando o *feedback* não encontra-se implementado, o circuito é chamado de loop aberto. Este tipo de configuração pode gerar um alto ganho mas também provoca uma alta instabilidade e, portanto, é raramente usado. Os amplificadores operacionais são empregados no condicionamento de sinais de diversos sensores, destacando os termopares, que são amplamente utilizados para medição de temperaturas para os mais diversos fins.

Considerando especificamente estes sensores, Desikan et al. (2016) realizaram estudos sobre o termopar tipo K, analisando a velocidade de resposta e alguns outros parâmetros. O termopar foi confeccionado especialmente para a análise, como mostrado na figura 3, e comparados com os outros modelos padronizados comercialmente. Além da análise da velocidade de resposta, foi verificada ainda a robustez desse termopar desenvolvido. O trabalho foi realizado utilizando uma junção do termopar na forma coaxial com contato de aproximadamente 2 mm no sentido axial e os testes foram realizados com água na temperatura de 49 °C a 100 °C. Também foram realizados testes para medição do ponto de estagnação em fluxo de calor, gerando um sinal de resposta que foi amplificado em 5000 vezes, possibilitando a leitura máxima do sinal no valor de 5 V. O termopar proposto se comportou de forma semelhante ao comercial e possibilitando um tempo de resposta de 3 μ s. A medição do fluxo de calor também apresentou boa concordância com valores teóricos. Outros parâmetros como robustez e possibilidade de adaptação à geometria de várias superfícies mostraram-se adequados e isto foi associado ainda a um baixo valor para a sua construção.

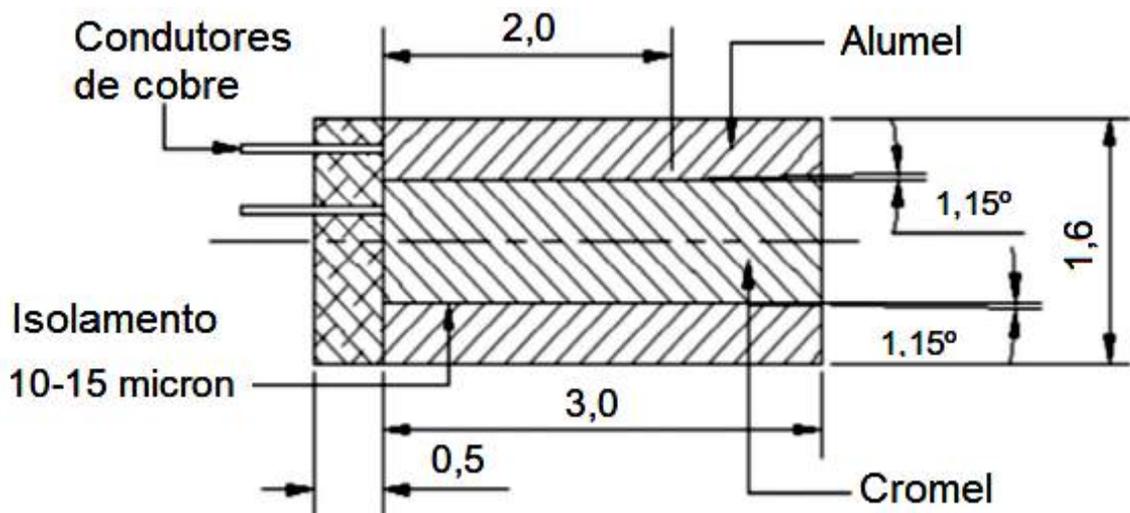


Figura 3: Esquema de um termopar coaxial. Adaptado de Desikan et al. , 2016

Outro dispositivo amplamente usado para medição de temperatura é o termistor, conforme mostrado por Bégin-Drolet et al (2013). Embora o trabalho tenha focado no desenvolvimento de anemômetros de canecas, necessitava-se de um dispositivo para medida de temperatura. Este controle era necessário para que o sistema pudesse atuar de forma a evitar a formação de gelo no equipamento. O uso de termistor tem uma grande vantagem em relação aos termopares, que é o fato de não precisar da utilização de um sinal de referência (zero eletrônico). Em contrapartida, a precisão destes sensores fica normalmente comprometida em função de uma resposta cujo comportamento é extremamente não linear.

Ainda com relação aos termistores, Rudtsch e Von Rohden (2015) demonstram a influência das variáveis na medição ultra precisa da temperatura ($<1\text{mK}$). Uma análise da calibração deles, também se verifica que os termistores se destacam por terem sensibilidade dez vezes maiores que um termômetro de resistência de platina, serem menos influenciados por choques mecânicos e vibrações e serem produzidos com dimensões pequenas. Os resultados do trabalho indicam a possibilidade de maximizar a resolução do dispositivo usando um resistor de referência com o valor próximo à máxima resistência do termistor. Usando o resistor com o valor similar ao nominal do termistor, verifica-se uma significativa redução do ruído e o desvio padrão é reduzido em 3 vezes quando comparado aos dados obtidos com o uso de resistores de referencia, com valores abaixo do nominal do termistor.

Outra variável amplamente mapeada em diversos setores é a velocidade angular e o posicionamento angular de equipamentos ou eixos girantes. Como exemplo, pode-se citar alguns sensores que usam este princípio: sensor de vazão, tacômetro, sensor velocidade e anemômetro. Em medições da velocidade angular, os sensores comumente utilizados são construídos usando o princípio magnético, como os de relutância variável ou de efeito Hall.

Kolhare e Thorat (2013) mostraram em seu trabalho o uso de medidores de vazão tipo turbina em aquecedores solares planos. A escolha deste medidor, entre os diversos modelos disponíveis no mercado, baseou-se na relação entre baixo custo e acuracidade, a qual pode atingir 0,1%. Neste trabalho também é destacado que o medidor de vazão tipo turbina pode ser facilmente adaptado ao uso em outros fluidos com diferentes viscosidades, necessitando somente o ajuste do fator de multiplicação, que pode ser obtido através da calibração do equipamento. Relacionado ao uso do medidor de vazão tipo turbina com diferentes fluidos, Dong-hui e Jing-yu, (2009) mostraram em seu trabalho que esse medidor pode ser usado com uma mistura de água e óleo, com a possibilidade de manter o erro relativo na faixa de $\pm 5\%$, desde que usado em conjunto com densitômetro de raios gama.

Um estudo de diferentes medidores de vazão foi realizado por Skea e Hall (1999), onde foram analisados medidores tipo turbina de pás reta e helicoidal, venturi, Coriolis e deslocamento positivo. Dados de um fluxo com fluido bifásico foram coletados por todos os medidores e mostraram que o medidor tipo turbina de pás retas e helicoidais apresentaram comportamento semelhante, com erro médio de $\pm 0,4\%$ e máximo de 1%. Os medidores tipo Venturi e Coriolis apresentaram alto erro nas condições de baixa vazão. O medidor de deslocamento positivo apresentou erro médio foi de $\pm 0,3\%$ e erro máximo de $\pm 0,6\%$. Com base no estudo, concluiu-se que os medidores que apresentaram maior confiabilidade para o caso estudado foram os do tipo turbina e deslocamento positivo.

Uma outra análise do medidor de vazão tipo turbina foi realizado por Rehman et al. (2011) e focada no estudo da melhor forma de obter e transmitir o sinal gerado pelo medidor. O sinal tipicamente elétrico do medidor de vazão foi substituído por sinal ótico e a transmissão tradicionalmente realizada por cabos elétricos foi realizada por cabos de fibra ótica. O comportamento do sinal ótico foi semelhante ao sinal elétrico, porém com a vantagem de poder ser transmitido a longa distância sem sofrer atenuação. Além disto, este tipo de sinal não foi afetado por interferências externas, tais como campo eletromagnético ou eletrostático

e poder ser usado com segurança com líquidos inflamáveis. O comportamento deste modelo de medidor foi semelhante ao tradicional e o erro máximo verificado foi da ordem de 0,4%.

Seguindo a mesma linha, Garmabdari et al. (2015) analisaram o sinal de equipamentos compostos por eixo girante. Fundamentalmente, o sinal estudado era de natureza elétrica e gerado por sensor de efeito Hall, muito aplicado em medidores de vazão tipo turbina. Após algumas análises, o trabalho sugere a utilização de medidores de vazão equipado por sensores tipo Hall utilizando uma interface do circuito eletrônico protegida dos efeitos de perturbação do sinal, como campos eletromagnéticos e temperatura.

Desta forma, Garmabdari et al. (2015) chama atenção que nos *encoders* tradicionais a precisão é baixa, devido a capacidade de medir somente a rotação completa. Diante desse fato, o estudo propõe um modelo que permitiria uma maior precisão em relação aos tradicionais, com imunidade contra perturbações magnéticas e capacidade de determinar o sentido do fluxo no escoamento.

Um modelo mais complexo usando o sensor Hall foi apresentado por Banjevic et al. (2012) e tem por finalidade medir a velocidade angular. Este novo dispositivo é composto de dois dispositivos Hall com 8 contatos circulares verticais (CVHD). No dispositivo proposto, o primeiro sensor é responsável pela detecção da amplitude da voltagem referente à rotação e o segundo pela contagem das voltas. A utilização de dois conjuntos de 8 CVHD para medição angular, dobra a sua sensibilidade, entretanto, reduz a sua escala de medição devido ao maior número de pulsos gerados. Ainda assim, esta forma de implementação pode ser compensada pela sua interface eletrônica e condicionamento de sinal, tornando possível a comparação desse sensor em termos de precisão com sensores de baixa escala. Este modelo de sensor permite a operação com velocidade angular na faixa de 628.000 rad/s, com erro de 4°, tempo de resposta de 1 μ s e resolução angular de 0,5°.

Outro equipamento para medida de rotação de eixo girante elaborado com o uso de sensor de efeito Hall é o anemômetro de caneca. Este tipo de dispositivo é muito usado para obter a velocidade média do vento devido a sua simplicidade, robustez aliado ao fato que não sofre influência da poluição do ar e depósitos de poeira.

Pindado et al. (2015) apresentaram um trabalho objetivando a compreensão do comportamento dos anemômetros de caneca e a melhoria do seu desempenho. Diferente dos

anemômetros comerciais, que tem o seu desempenho baseado em uma função do primeiro grau, foi desenvolvido um modelo analítico para representar o movimento do anemômetro de 3 canecas durante a rotação completa do aparelho. Este novo equacionamento pode ser dividido em uma constante e duas harmônicas. As harmônicas representam os efeitos físicos e a influência da geometria da caneca no aparelho. O trabalho destaca ainda que as harmônicas podem ser consideradas como uma decomposição espectral e usadas para avaliar o desempenho do anemômetro. A alteração dessas harmônicas, principalmente a primeira, indica possíveis avarias no equipamento.

Em outro trabalho também realizado por Pindado et al. (2012), foi destacado o uso dos anemômetros no setor de energia eólica e, entre eles, o anemômetro de caneca. No estudo demonstrou-se que este instrumento é importante nestas análises devido à sua linearidade, precisão e rápido tempo de resposta a uma eventual aceleração e desaceleração do vento. Neste trabalho foram analisados os coeficientes de calibração da função transferência do anemômetro, que é uma equação do primeiro grau, composta de uma constante isolada e outra constante que multiplica a frequência ($V = A_r \cdot f + B$). Foi demonstrado que ambas as constantes têm influência do raio do centro de rotação. A constante A_r tem relação direta com a aerodinâmica e com a área frontal do copo. A constante B , por outro lado, é influenciada diretamente apenas pela área frontal do copo.

Considerando o amplo espectro de trabalhos voltados para a medida de grandezas fundamentais e o alto custo de sua aquisição, o presente trabalho objetiva elaborar um módulo de coleta de dados de baixo custo. Para este fim, será usada uma placa de microcontrolador identificada como *Arduino Uno* para processar as informações fornecidas pelos sensores e armazenar os dados em um cartão SD (Secure Digital Card), facilitando a sua exportação. Esse módulo deverá obter informações de parâmetros como: velocidade do vento, direção do vento, temperatura por termopares, temperatura por termistor, temperatura por sensor digital, vazão, data e hora da coleta dos dados, auxiliando em diversas atividades de instrumentação e controle em aplicações na engenharia.

III - MATERIAIS E MÉTODOS

O módulo de coleta de dados desenvolvido neste trabalho pretende integrar os principais sensores usados no mais diversos estudos de equipamentos, embora o foco seja direcionado para avaliação de coletores solares. Para tanto, existe a possibilidade de inclusão de sensores específicos de acordo com a necessidade de coleta de dados de cada modelo de equipamento. Os sensores que serão coletados neste trabalho são tipicamente empregados em diversas pesquisas e, alguns deles inclusive, são comumente encontrados no setor industrial para monitoramento das variáveis de produção.

Existem alguns sensores que necessitam de condicionadores de sinais, como conversores e amplificadores operacionais, para poderem integrar-se ao sistema de coleta de dados, como a placa de microcontrolador Arduino. Nestes casos, o sinal elétrico gerado é de baixa amplitude, o que impossibilita a sua aquisição e interpretação direta pelo microprocessador. Nestes casos, estes dispositivos devem ser construídos em separado para permitir a interpretação do sinal pelo sistema de controle ou registro do sinal.

Com o sinal corretamente obtido, é possível o seu tratamento, manipulação das informações, o armazenamento dos dados e a comunicação com o usuário. Estes aspectos estão diretamente relacionadas à programação do módulo. Ela deve ser elaborada para o uso dedicado à cada sensor, tornando o equipamento de coleta de dados modular, além de evitar o conflito ou interferência de outros sensores quando usados no mesmo módulo. Neste capítulo serão apresentados diversos grupos de sensores e seus princípios de operação, de forma a facilitar a elaboração de uma biblioteca geral.

III.1. Sensores Hall

Os dispositivos de efeito Hall tem a característica de gerar uma diferença de potencial proporcional à intensidade do campo magnético o qual está inserido, porém, o nível deste sinal gerado é muito baixo, necessitando usar amplificadores para elevar o seu valor. Os sensores de efeito Hall disponíveis comercialmente são normalmente compostos pelo sensor propriamente dito e um circuito integrado (CI) amplificador. Este conjunto encontra-se normalmente protegido por um invólucro, identificado como encapsulamento, tornando-os imunes à poeira, lama e água. A robustez advinda desta característica aumenta as vantagens do seu uso sobre os sensores óticos e eletromecânicos. Entre os sensores de efeito Hall comerciais há dois modelos que se destacam, o que fornece o sinal analógico proporcional ao fluxo magnético e o que fornece sinal digital quando interceptado por campo magnético de intensidade mínima fixado pelo fabricante do sensor. Esta diversidade aumenta as possibilidades de sua aplicação em diferentes contextos, com destaque para os equipamentos que possuem eixo girante, como medidores de rotação em componentes automotivos, em dentes de engrenagens, indicadores de velocidades, sensores de fluxo de fluidos, etc.

III.2. Sensor de Vazão

Embora existam diversas possibilidades para medida de vazão, o medidor do tipo turbina é, sem dúvida, um dos mais utilizados. Esta classe de medidores têm o seu funcionamento baseado na geração de um pulso elétrico gerado por um sensor magnético durante a passagem das pás do rotor próximo a este sensor. Assim sendo, o número de pulsos por rotação é dependente do número de pás do rotor e a frequência proporcional a rotação do seu eixo.

Com o intuito de caracterizar o funcionamento deste sensor e o sinal elétrico gerado por ele, foi realizada a sua análise em funcionamento com o auxílio do osciloscópio modelo MO-2061 com resposta em frequência de 60 MHz, fabricante Minipa, conforme figura 4. O sensor de vazão modelo YF-S201, fabricante SEA, foi acoplado ao canal 1 do osciloscópio, o qual teve o ajuste da grade com a tensão por divisões em 2 volts e o tempo por divisão em 5 ms, resultando na visualização do sinal em forma de uma onda quadrada, com pouco interferência de sinais indesejáveis, sendo a frequência deste sinal proporcional a vazão.

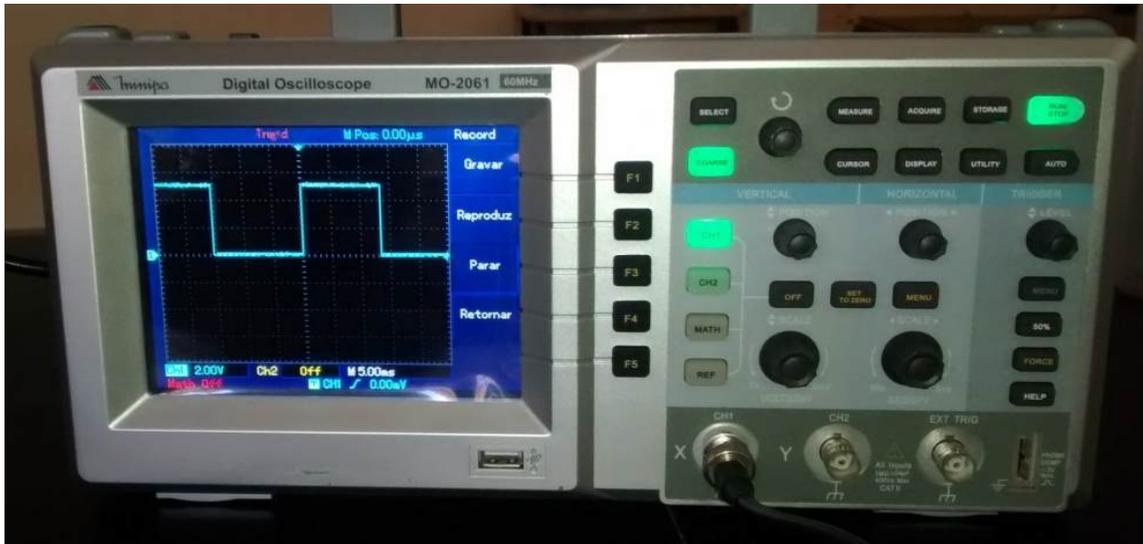


Figura 4: Análise do sinal gerado pelo sensor de vazão.

Conhecido o comportamento do sinal do sensor e a informação fornecida pelo fabricante da razão entre a frequência dos pulsos e a vazão, é possível realizar as suas medidas. No caso analisado, o medidor estabelecia a proporção de um sétimo da vazão em litros por minutos (frequência= vazão/7), desde que aplicada vazão mínima de 1L/min e máxima de 30 L/min. Este é o conjunto de informações necessárias para realizar a programação do sensor de vazão e posteriormente acoplá-lo ao módulo de recepção e processamento de sinal.

No intuito de evitar a presença de ruídos no sinal gerado pelo sensor, incorporou-se um resistor de 10 K Ω em paralelo com a alimentação 5V e a saída do sinal, conforme mostrado na figura 5. A utilização deste resistor torna o sinal gerado pelo sensor mais estável e evita erros de interpretação pelo receptor. O diagrama elétrico da montagem do dispositivo está mostrado na figura 6.

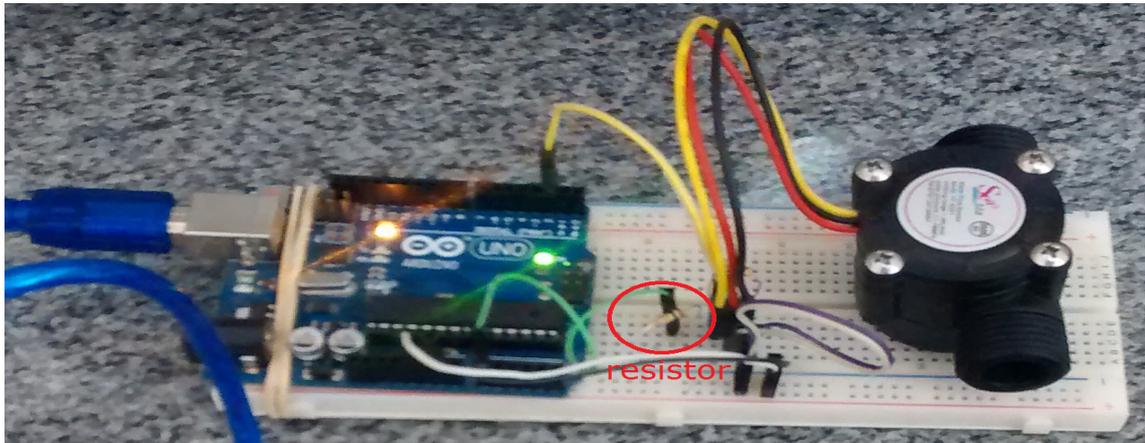


Figura 5: Módulo de processamento de sinal, Arduino, com o resistor Pull Up e sensor de vazão durante os testes da programação.

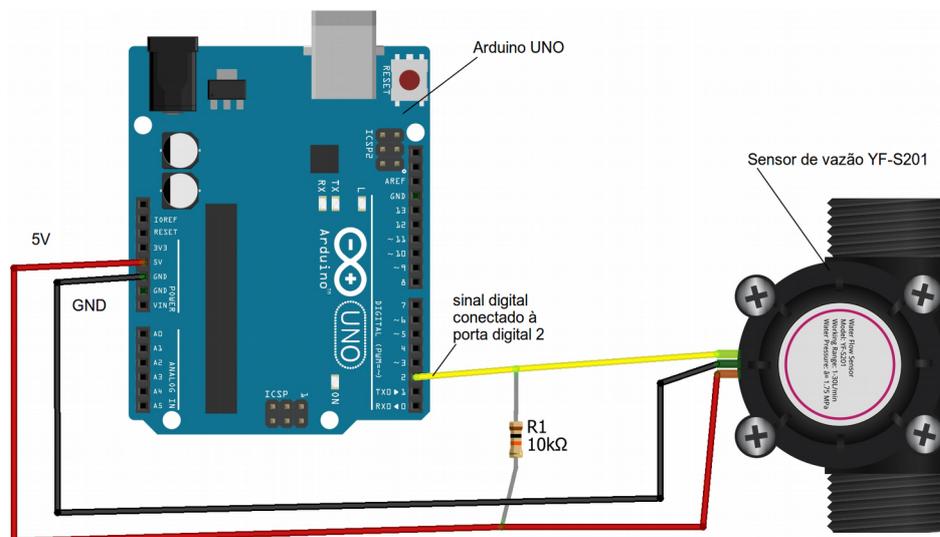


Figura 6: Esquema elétrico para acoplar o sensor de vazão ao processador de sinal.

III.2.1. Programação sensor de vazão

A função definida na programação do processador de sinal para fazer a captura do sinal provindo do sensor de vazão foi:

attachInterrupt (pin, ISR, mode) ;

sendo:

- *pin* é o pino do Arduino que recebe o sinal;
- *ISR* é a função a ser executada quando o sinal é recebido;
- *Mode* é quando o sinal será computado, pode ser usado 4 tipo de *mode*, sendo eles:
 - *LOW*: acionado quando o sinal estiver em baixo;
 - *CHANGE*: acionado quando o sinal mudar de valor alto para baixo ou baixo para alto;
 - *RISING*: acionado quando o sinal sair de baixo para alto;
 - *FALLING*: acionado quando o sinal for de alto para baixo;

O sinal do sensor de vazão gera uma onda quadrada, conforme figura 4, possibilita o uso da função interrupção tanto com a subida como descida do sinal. Neste trabalho adotou-se o uso do *RISING*, que chama a função definida quando o valor do sinal se elevar, isto é, quando a pá do rotor do sensor de vazão adentrar na região sensível do seu sensor magnético.

Definindo a função ISR

Inicialmente, foi concebida uma programação que se baseava no comando *attachInterrupt(pin, ISR, mode)* na estrutura *void setup()* do programa. Este procedimento deixa o comando ativo durante a execução de toda a programação de acordo com a estrutura:

```

/*-----sensor vazão-----*/

volatile float contador; //armazena o a subida do sinal (hall)
float Calc;
int hallsensor = 2; //pino conectado ao sensor ( digital 0)
int pinohall = 0; //pino conectado ao sensor com identificação digital

void rpm () //função que conta os pulsos do sensor hall
{
  contador++;
}

void setup()
{
  Serial.begin(9600); //ativa e ajusta a velocidade de comunicação,
  pinMode(hallsensor, INPUT); //ativa o pino 2 como entrada de sinal
  attachInterrupt(pinohall, rpm, RISING); /*ativa a função de interrupção
  no pino 2 - pinohall (digital 0),
  executa a função rpm quando o sinal vai de baixo para alto*/
}

```

```

void loop ()
{
  contador = 0; //zera a variavel
  sei(); //ativa a função interrupção
  delay (1000);
  cli(); //desativa a função interrupção
  Calc = (contador * 60 / 7); //(Pulse frequency*60)/7xQ=flow rate in L/hour
  Serial.print (Calc,DEC); //imprime o valor calculado acima
  Serial.print (" L/hour"); //imprime "L/hour"
}

```

Esta programação mostrou-se adequada para este modelo de sensor por ser dedicada, com programação curta, ocupando pouca memória e sendo de rápida execução. A sua execução se resume á três operações, conforme representada na figura 7: contagem do número de pulsos em 1 segundo, calculo da vazão e transmitir a informação ao usuário do valor obtido.

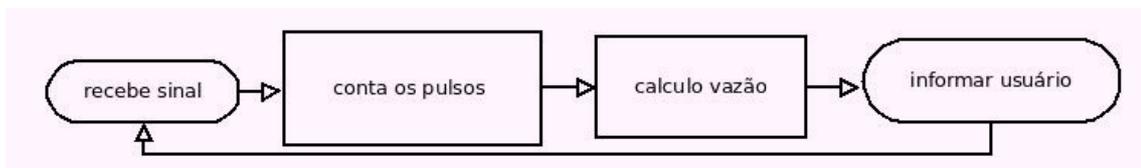


Figura 7: Fluxograma da programação sensor de vazão.

Com o objetivo de melhorar a precisão da leitura, a programação foi alterada deixando de realizar a leitura da quantidade de pulsos no intervalo definido e passando a determinar o intervalo de tempo entre os pulsos. Este procedimento pode evitar uma série de imprecisões como, por exemplo, um sensor de vazão que tenha a geração de um pulso a cada quarto de volta e no intervalo de um segundo tiver realizado o movimento de 2 voltas e 1/8. Neste caso, ele geraria apenas 4 pulsos, não computando 1/8 de volta, e a estimativa da vazão seria baseada numa quantidade errônea de pulsos, gerando imprecisão na leitura.

Nesta reelaboração da programação foi inserido o comando *millis()*, que faz a contagem do tempo em milésimo de segundos. Desta forma, foi possível a determinação do intervalo de tempo entre os pulsos ou entre a passagem das pás do rotor ao elemento sensível. Com esse valor calcula-se a frequência do sinal e, posteriormente, a vazão percorrendo o medidor. Esta nova estrutura de programação está representado através de um fluxograma mostrado na figura 8.

Porém o uso do comando *millis()* juntamente com a função *attachInterrupt(pin, ISR, mode)* pode causar conflito e interromper a execução da programação. Como solução da integração deles, o comando *millis()* não deve estar na fase de *looping* da programação e deve ser introduzido em uma função em separado, identificada neste trabalho como *timepulso()*. Esta função é chamada ao longo da programação para ser executada, interrompendo o *looping* e o retomando após a sua execução. Nesta arquitetura de código, o *looping* retorna sem gerar atrasos ou parada inesperada na programação.

A programação completa está disponível no apêndice A, porém a função que computa o tempo entre os dois pulsos, *timepulso()*, está apresentada abaixo e exemplificada através do fluxograma da figura 8.

```
void timepulso()
{
  contador = 0; //zera a variável de registro, declarada no início da programação
  totalTime = 0; //zera o tempo entre o primeiro e o n pulso,
  sei(); //ativa a função interrupção que está em void setup()
  if (contador == 1) //inicia a contagem quando receber o primeiro sinal
  {
    startTime = millis();
    float tempolimite;
    while (contador < 5) //fica em loop até receber 5 sinais do sensor hall
    {
      tempolimite=millis()-startTime; //calcula o tempo entre o primeiro e quinto sinal
      if (tempolimite>5000) //caso não tenha 5 sinais em 5 segundos, aborda a contagem
      {contador=0; break;}
    }
    totalTime=(millis()-startTime)/4; /*faz a média do tempo para intervalo entre 2
    sinais*/
  }
  cli();
}
```

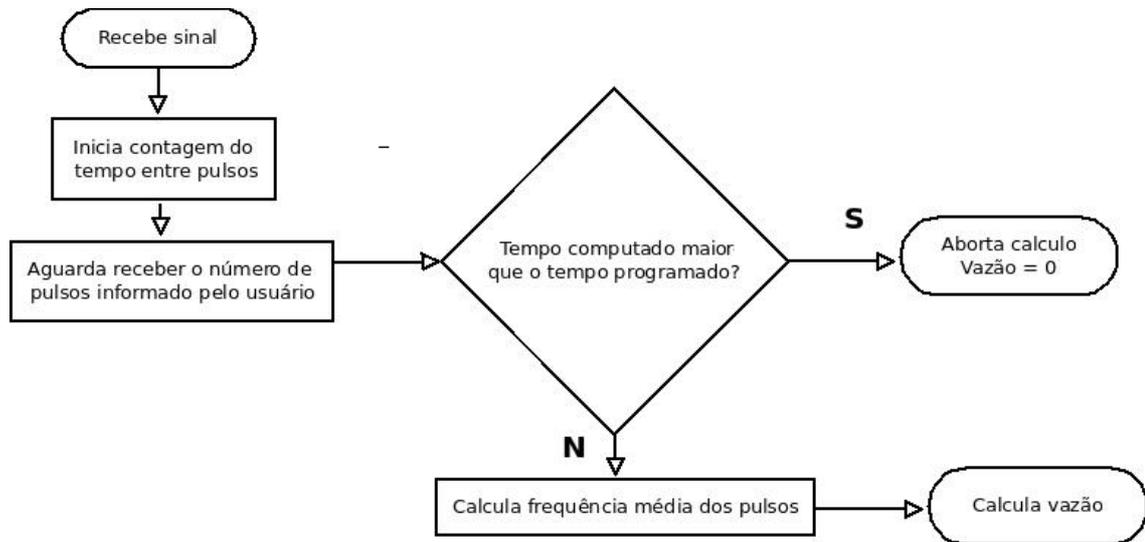


Figura 8: Fluxograma da função programação com a contagem do tempo entre os pulsos.

No fluxograma da figura 8 nota-se que há uma tomada de decisão com relação ao tempo computado. Esta função tem a finalidade de evitar que a execução do programa fique pausada aguardando receber a quantidade de sinais definida pelo usuário. Deste modo, caso o módulo de coleta de dados não receba a quantidade de pulsos programada no tempo limite definido, o que pode ocorrer caso a vazão cesse durante a medição, o programa considera que houve um erro na leitura da vazão e retorna para continuar a execução da rotina *looping*.

III.3. Anemômetro

Um anemômetro de 3 canecas foi construído e estudado durante o desenvolvimento deste trabalho. Este tipo de anemômetro pode ser equipado com dois tipos de sensores, o sensor tipo Hall ou sensor tipo Reed-Switch, ambos relacionados à rotação das canecas do equipamento.

Se o sensor usado no anemômetro for o de efeito Hall, semelhante ao usado no sensor de vazão, pode-se usar a mesma rotina de programação do sensor de vazão para computar o tempo entre dois pulsos. Cabe, entretanto destacar que o uso desta informação deve ser processada com base em uma função diferente, elaborada especificamente para o cálculo da velocidade do vento e com coeficientes definidos pela calibração do anemômetro.

Conforme trabalho realizado por Pindado et al. (2015), o qual avaliava o comportamento dos anemômetros de tipo caneca e possíveis melhorias do seu desempenho, os anemômetros comerciais tem o seu comportamento baseado na função:

$$V = A \cdot f + B \quad (1)$$

onde V é a velocidade do vento em m/s ou km/h dependendo dos coeficientes de calibração, f a frequência do sinal gerado pela rotação do equipamento em Hz, A e B são os coeficientes de calibração. No intuito de ter uma expressão que representasse de forma clara os fenômenos físicos, a equação acima foi reescrita em função da frequência de pulsos totais, f_r , introduzindo assim o número de pulsos por rotação do anemômetro, N_p .

$$V = A \cdot N_p \cdot f_r + B \quad (2)$$

Em anemômetros que tenham o sinal gerado com a característica de um pulso por volta, deve também possuir o tempo registrado entre dois pulsos. Este intervalo está diretamente relacionado à rotação de seu eixo e a equação 1 deve ser aplicada para definir a velocidade do vento. Caso o anemômetro tenha mais de um sensor ou mais de um pulso por volta, a rotação de seu eixo definida na equação 2 deve ser utilizada.

O anemômetro construído neste trabalho objetivou o teste do módulo de coleta de dados, sendo composto majoritariamente por material plástico PLA (ácido polilático). Material que proporciona leveza, resistência mecânica e resistência as ações climáticas. O projeto original do anemômetro indicava que ele seria composto por 3 canecas com diâmetro interno de 45mm e raio de giro de 85mm, conforme ilustrado na figura 9 e 10.

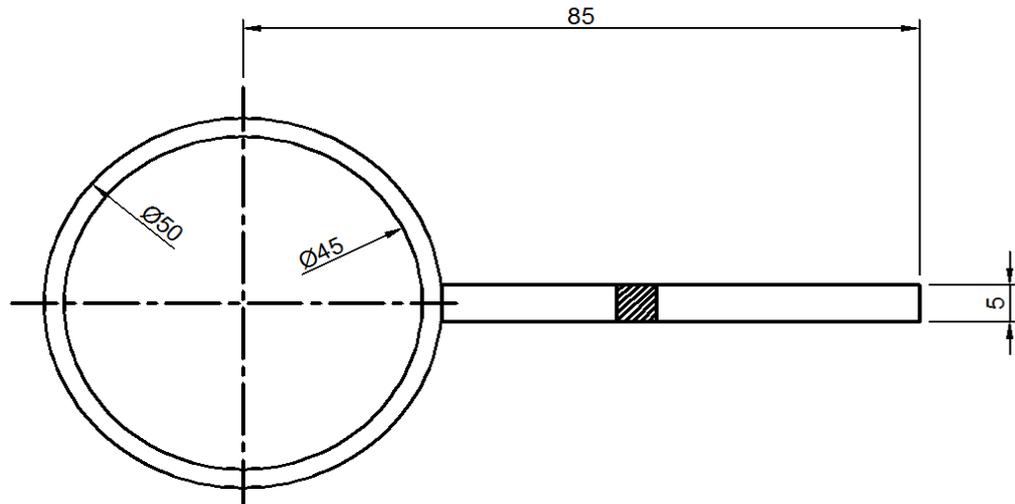


Figura 9: Dimensões das canecas do anemômetro.



Figura 10: Anemômetro de 3 canecas com sensor tipo Hall e três acionadores do sensor por volta.

Para a geração do sinal e mapeamento da rotação do eixo do equipamento, utiliza-se um sensor de efeito Hall. Para alojar este sensor, o qual deve estar afastado do ímã a uma distância aproximada de 5 mm, foi desenvolvido um alojamento, também de material plástico PLA, que será fixado no eixo base do anemômetro, conforme esquema mostrado na figura 11.

O acionamento do sensor de efeito Hall é realizado por 3 ímãs de seção circular e diâmetro 8 mm, de material neodímio, posicionados de forma equidistante, ilustrado na figura 12, fornecendo a característica do equipamento de ter 3 pulsos por volta.



Figura 11: Sensor de efeito Hall inserido no alojamento e desenho do alojamento na fase de projeto.

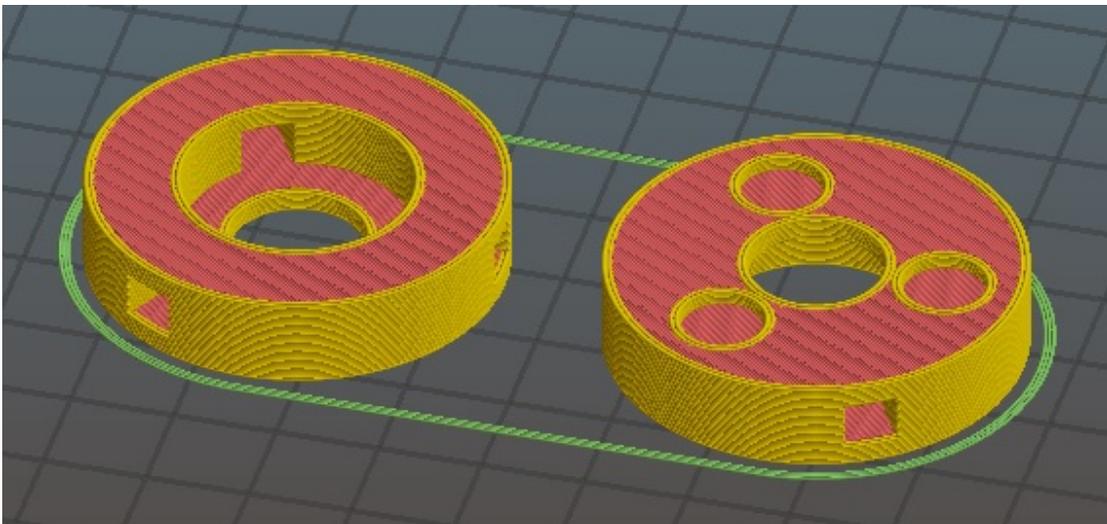


Figura 12: Alojamento dos ímãs e rolamento – fase de projeto.

Seguindo o mesmo princípio utilizado no sensor de vazão (sensor tipo Hall), deve-se ter na montagem do circuito eletrônico um resistor de $10\text{ K}\Omega$ em paralelo com a saída do sinal e a alimentação 5 V . No entanto, caso o anemômetro utilize-se de sensor tipo Reed Switch, um novo código deve ser elaborado. Este tipo de sensor tem a característica de uma chave acionada pelo campo magnético. Ele é composto por um bulbo de vidro com dois contatos no

seu interior, separados por uma pequena distância e o acionamento ocorre pelo movimento dos contatos. Neste caso o esquema elétrico também deve ser alterado e o uso do resistor de $10\text{ K}\Omega$ é feito em paralelo com a saída do sinal e o *ground* (terra) do arduino, conforme ilustrado na figura 13. Este procedimento assegura a estabilidade do sinal, uma vez que este tipo de contato pode gerar oscilação na abertura ou repique.

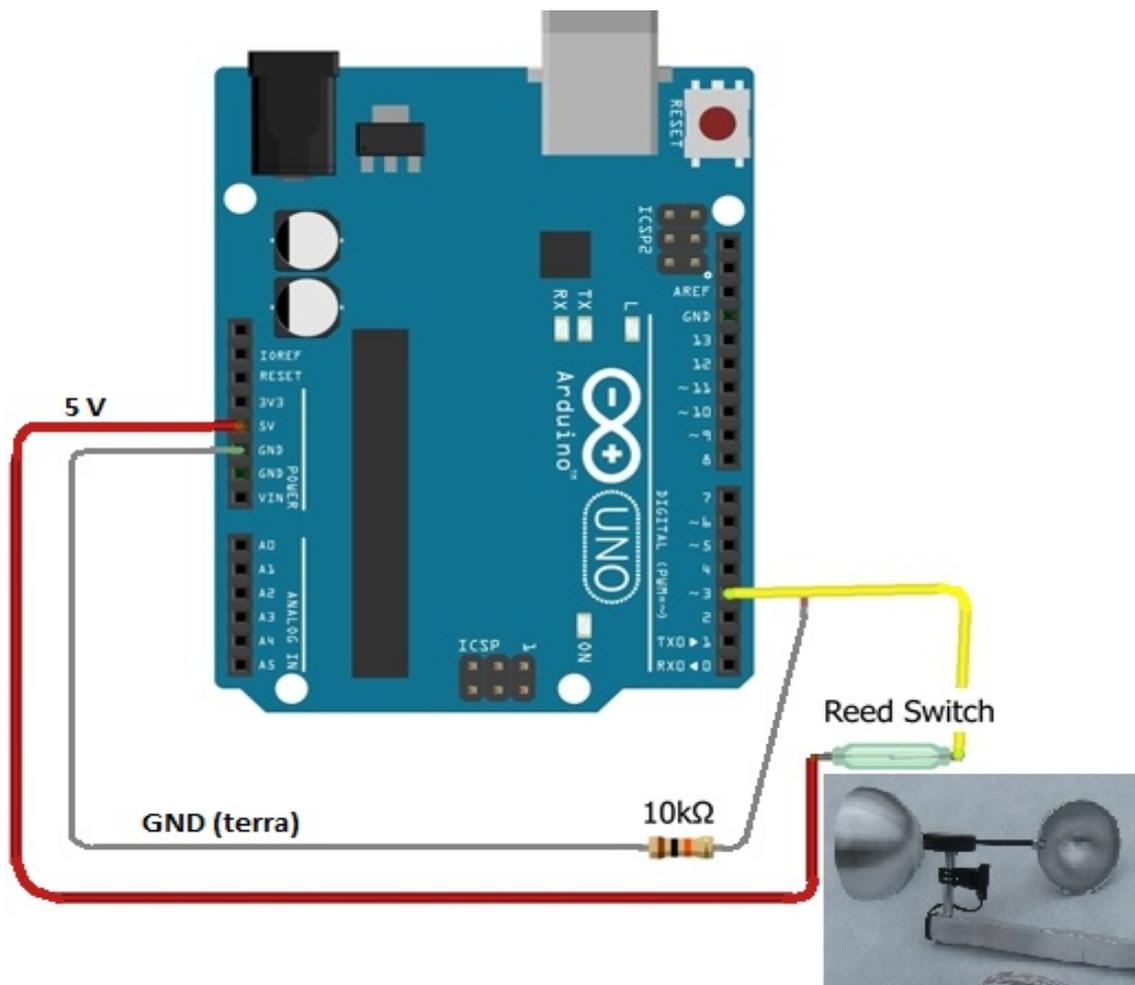


Figura 13: Esquema elétrico para acoplar o anemômetro com sensor Reed Switch ao processador de sinal.

III.3.1. Programação anemômetro

A programação do anemômetro é similar à dos equipamentos que usam o sensor de efeito Hall para obtenção do sinal, isto é, a interrupção externa do programa para computar o sinal que sofreu alteração de estado, de baixo para alto. Entretanto nesse caso existe a

peculiaridade do sinal alterar o seu estado frequentemente em função do movimento das canecas, interrompendo a execução da programação de forma muito mais contínua.

Caso o instrumento esteja sendo utilizado somente para o anemômetro, sem outros sensores conectados ao módulo de coleta de dados, essa frequente interrupção na programação não afeta o seu desempenho. Porém, caso ele esteja sendo utilizado em conjunto com outros sensores no módulo, deve-se estabelecer um tempo para a realização da medida e posteriormente desabilitar a interrupção externa para o recebimento do sinal. Na programação proposta, a captação do sinal do anemômetro será realizada uma vez a cada *loop* do programa e, com isso, permitirá a obtenção dos dados dos demais sensores de forma sequencial.

Um modelo de programação dedicada ao anemômetro semelhante ao mostrado na figura 10, com a consideração do tempo limite de leitura e cálculo médio da frequência dos sinais está exemplificada abaixo e a sua lógica de operação está demonstrada na figura 14, através de um fluxograma.

```
// Constantes a serem definidas na calibração
float A=1.6736;//variável A da função v=Ax+B(Np=1) ou v=A.Np.Fr+B
float B = 0.9984;//variável B da função v=Ax+B do anemômetro
float tempolimit = 4000; /*tempo limite para medição,definido em função
da menor velocidade do vento que o equipamento registra*/

// Declaração das Variáveis da programação

volatile int contador=0UL;
float velocidade = 0; // velocidade do vento
int porta =3; //pino ligado ao anemômetro
unsigned long totalTime=0UL;

void contapulso () //função que conta os pulsos do sensor hall
{
  contador++;
}

void setup()
{
  pinMode(porta, INPUT); // pino do anemômetro
  /*função de interrupção no pino 3 - (digital 1), executa a função
  rpm quando o sinal vai de baixo para alto */
  attachInterrupt(digitalPinToInterrupt(porta),contapulso, RISING);
  Serial.begin(9600); // ativa a comunicação serial
}

void loop()
{
  Serial.print ("veloc: ");
  calcvelocidade();//calcula a velocidade do vento
  Serial.print (velocidade);
}
```

```

Serial.println (" m/s");
Serial.print(" tempo de uma rotacao: ");
Serial.print (totalTime);
Serial.println (" ms");
delay (10000); //tempo para iniciar nova leitura-10s
}
/*-----função calcula tempo entre pulsos-----*/
void calcvelocidade ()
{
    totalTime = 0; //tempo entre os pulsos
    float startTime = 0;
    float tempo = 0;
    sei(); //ativa a função interrupção - attachInterrupt(pino,contapulso,
RISING);
    startTime = millis();
    if (contador == 0)
    {
        while (contador == 0) //espera 2 seg para receber algum sinal
        //util para baixas velocidades
        {
            tempo = millis()-startTime; //calcula o tempo de inicio da rotina
            if (tempo > 2000) //caso não tenha sinal em 2 seg, aborta
                {contador=0; break;}
        }
    }

    if (contador == 1) //inicia a contagem quando receber o primeiro sinal
    {
        startTime = millis();
        while (contador < 4) //fica em loop até receber +3 sinais do sensor hall
        {
            tempo = millis()-startTime; //calcula o intervalo entre o primeiro e
            //sinal
            if (tempo > tempolimite) //caso não tenha 3 sinais no tempo determinado,
            //aborda a contagem
                {contador=0; break;}
        }
        detachInterrupt(digitalPinToInterrupt(porta));
        if (tempo <= tempolimite)
        {
            totalTime = (millis()-startTime)/3; /* faz a média do tempo para
            intervalo entre Np sinais-fixo Np=3 neste exemplo, mas na biblioteca disponível
            no apêndice, Np deve ser informado*/
            contador = 0;
        }
    }

    if (contador != 0) {contador = 0;}
    float frequencia;
    if (totalTime > 0)
    {
        frequencia = (1/totalTime)*1000;
        velocidade = (frequencia * A + B);
        if (velocidade < 0) // caso ocorra algum erro de calculo
            {velocidade=888888888;}
    }
    else
    {
        velocidade = 0;
    }
}

```

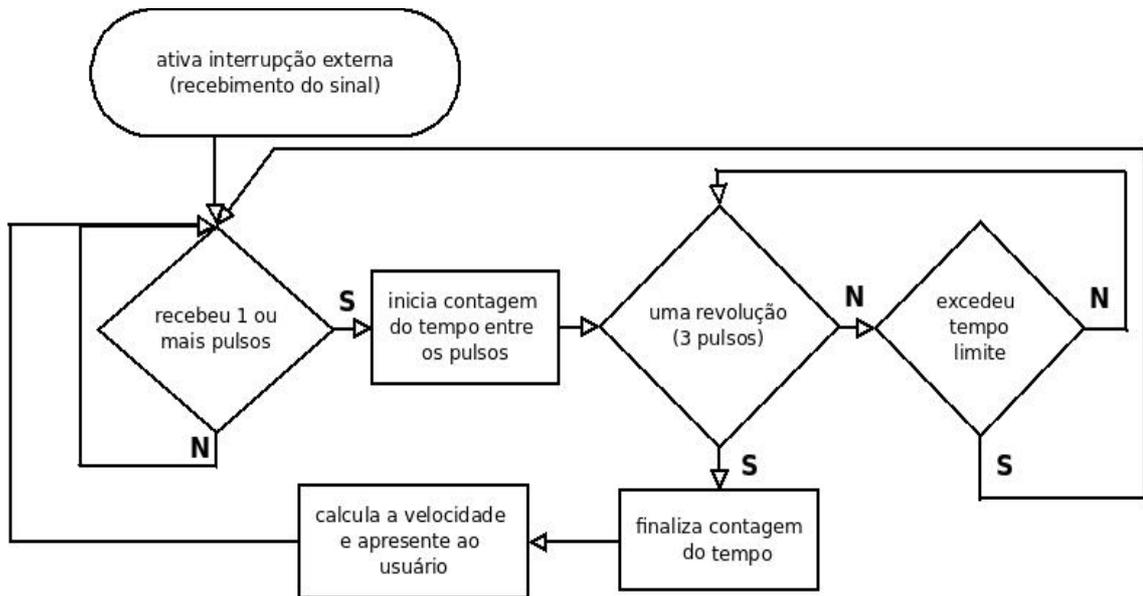


Figura 14: Fluxograma da programação do anemômetro de 3 canecas com 3 pulsos por volta.

III.4. Direção do Vento

Assim como a determinação da velocidade do vento, a direção do vento também é um parâmetro importante na avaliação das condições ambientais. Pensando nisto, foi montado também um dispositivo capaz de medir a direção do vento. Foi utilizado na montagem do dispositivo um leme que acompanha a direção do vento e, acoplado a ele, um sensor que indique o seu posicionamento. Também neste caso existem algumas alternativas para a escolha do sensor a ser usado. Dentre estas alternativas, os potenciômetros, os encoders elétricos e óticos, os Reed Switch associado a resistores e os sensores de efeito Hall são os de mais fácil implementação.

Inicialmente foi projetada a estrutura física do dispositivo e, posteriormente, o mesmo foi construído usando material plástico PLA (ácido polilático). Este material tem propriedades semelhantes ao plástico PET e deriva de materiais orgânicos de fontes renováveis. Esse material foi escolhido pelo seu baixo peso específico e pela possibilidade de construção do protótipo através de uma impressora 3D, garantindo assim a fidelidade da geometria projetada, rapidez e baixo custo de confecção.

Com o propósito de deixar a estrutura mais leve e com menor inércia para o deslocamento, as peças foram impressas utilizando um geometria honeycomb no seu interior, com preenchimento de 25%, conforme apresentado na figura 15.

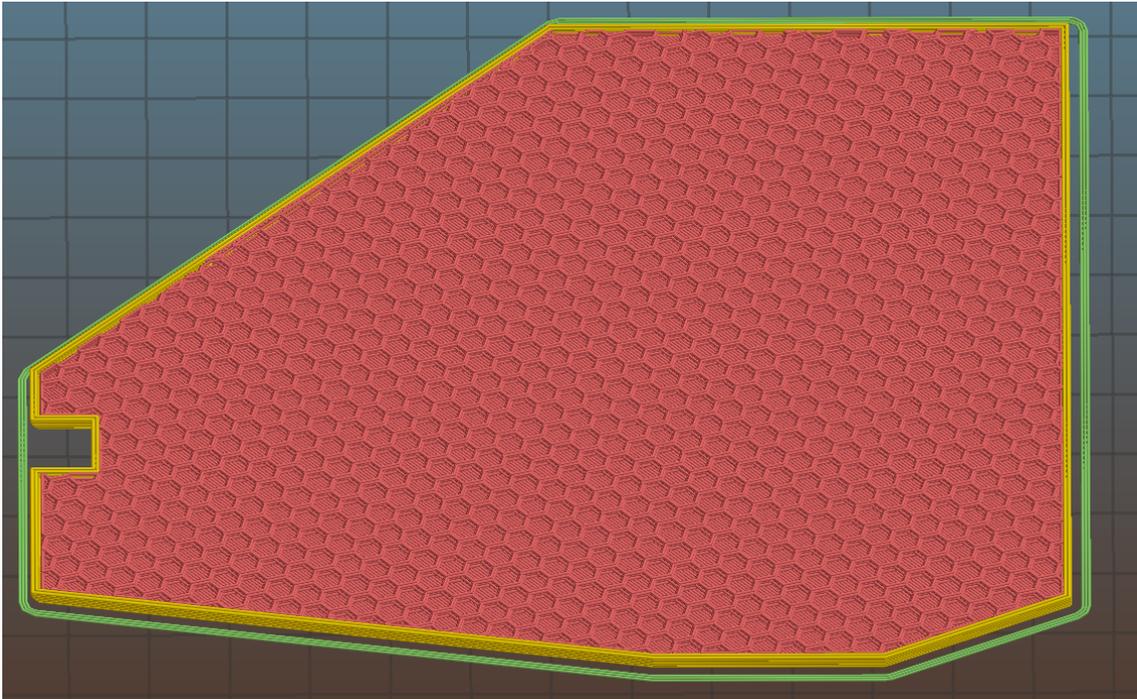


Figura 15: Desenho do leme com destaque da sua estrutura interna de honeycomb.

Para o monitoramento do movimento do leme foi utilizado como sensor um encoder mecânico mostrado na figura 16, com as seguintes características:

- Pulsos por revolução : 25;
- Tensão de Alimentação: 5V;
- Rotação: 360° (Sem limite);
- 2 Saídas de Pulsos defasadas 90°, possibilitando a leitura do sentido de giro. (Código Gray);
- Dimensões: Largura: 17mm e comprimento: 37mm



Figura 16: Encoder multivolts.

O encoder apresentado é um componente indicado para uso interno sem umidade e, portanto, não suporta condições com umidade, passível de ambientes externos. Para contornar este problema, foi elaborado um alojamento para o encoder onde ele foi fixado e selado com resina evitando, assim, que a umidade danifique o dispositivo. Uma foto e um esquema deste modelo de invólucro podem ser vistos na figura 17.

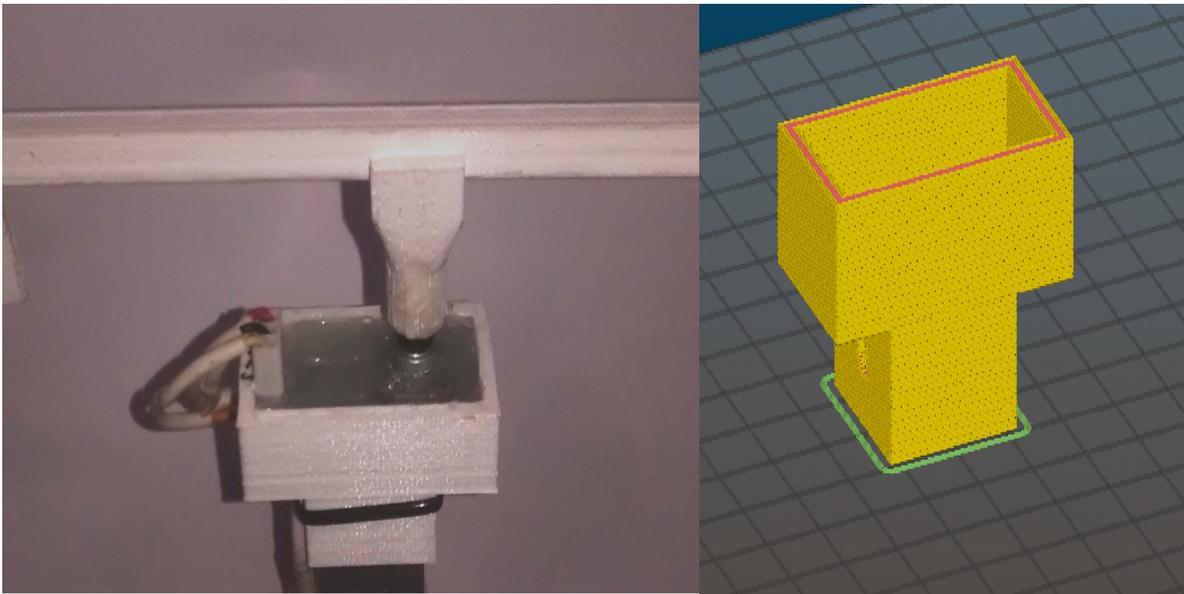


Figura 17: Alojamento do encoder, situado na base do equipamento.

A montagem do dispositivo ocorre com a união de todos os seus componentes: o alojamento do sensor, o eixo do leme, o leme e contrapeso do leme para balancear o sistema.

O equipamento foi fixado no topo de uma barra de alumínio perfilado, de forma que o anemômetro pudesse acompanhá-lo na mesma estrutura, conforme mostrado na figura 18.

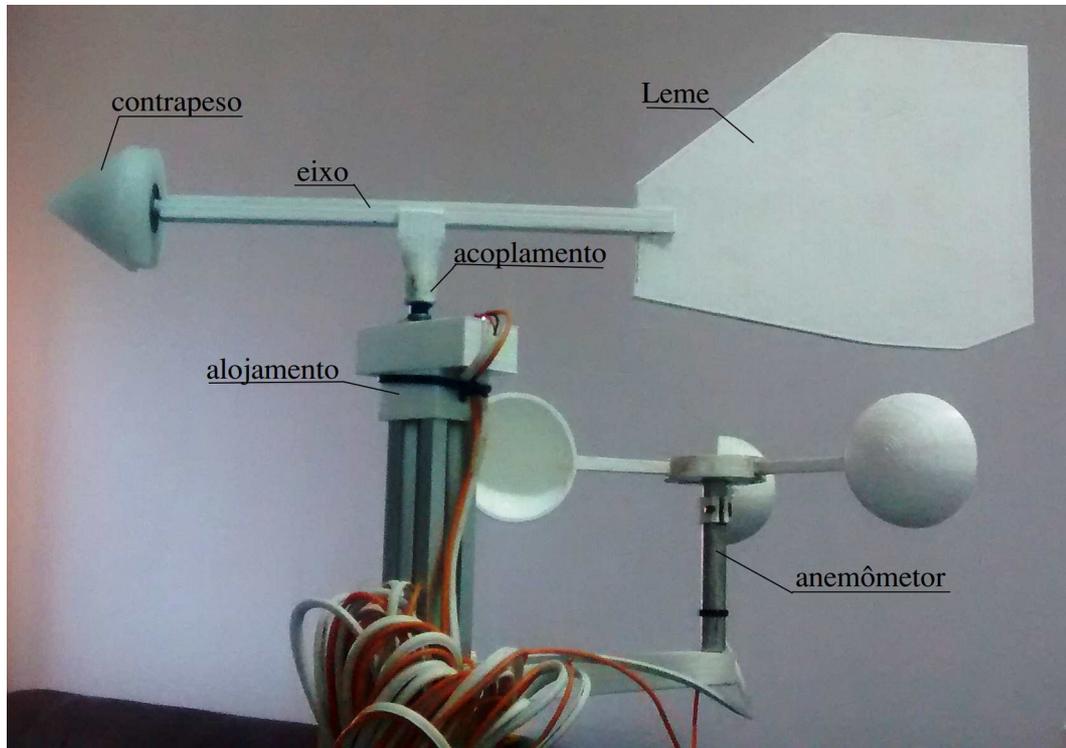


Figura 18: Sistema para medição da velocidade e posição do vento.

Por se tratar de um encoder incremental, o sensor utilizado precisa ter registrada uma posição conhecida, por exemplo o sentido norte. Definida a posição de referência, a cada movimento do leme será gerado um sinal nível alto o qual o software irá interpretar como a mudança de posição e a atualizará em função da rotação em graus.

III.4.1. Programação do sensor direção do vento

De acordo com o exposto, o encoder usado nesse sensor é conectado ao módulo de leitura por 3 conectores, sendo um dedicado a alimentação de 5V e os outros dois fornecem o sinal em nível alto e baixo. A combinação desses sinais gera um código de 2 bits (código Gray), tornando possível a identificação da alteração da posição do sensor e o seu sentido de giro. O código de 2 bits tem 4 combinações possíveis, sendo elas em binário 00, 01, 10 e 11. Para a identificação do sentido de giro é necessário comparar a posição anterior com a nova, por exemplo, se o sensor está na posição 01 e a nova posição será 00, haverá o movimentou

no sentido anti-horário ou caso a posição for 10, o movimentado será no sentido horário. Este movimento gera uma combinação de duas posições, como 0100, que significa que saiu da posição 01 e está na posição 00, portanto em movimento anti-horário.

A interpretação do código Gray é realizado durante a execução da programação pela biblioteca *RotaryEncoderV1.h*, a sua utilização pode ser verificada na programação abaixo, elaborada para identifica a posição do sensor em graus e a sua lógica de operação é mostrada no fluxograma da figura 19.

```
#include <RotaryEncoderV1.h>

RotaryEncoderV1 encoder(A2,A3); //Setup RotaryEncoderV1 para os pinos A2 and A3
// sinal 1=> pino A3 sinal 2=> pino A2 do arduino UNO
void setup()
{
  Serial.begin(9600); //inicia a comunicação serial
  Serial.println("Uso do encoder P17 como sensor de posição em graus.");

  /*Caso queira usar outros pinos do arduino UNO que não sejam o A2 e A3, modifique
  as 2 linhas abaixo*/
  PCICR |= (1 << PCIE1); /* Ativa interrupção do port C que inclui os pinos
  analógicos*/
  PCMSK1 |= (1 << PCINT10) | (1 << PCINT11); //Ativa interrupção pinos A2 e A3
}

void loop()
{
  static int pos = 0;
  int newPos = encoder.getPosition(); //verifica a posição do sensor
  if (pos != newPos) //caso a posição tenha sido alterada informa o usuário
  {
    Serial.print(newPos);
    Serial.println();
    pos = newPos;
  }
}

ISR(PCINT1_vect) // "chama" a rotina tick quando ocorre alteração do sinal nos pinos
//A2 e A3
{ encoder.tick(); } //a rotina tick está na biblioteca RotaryEncoderV1.cpp
```

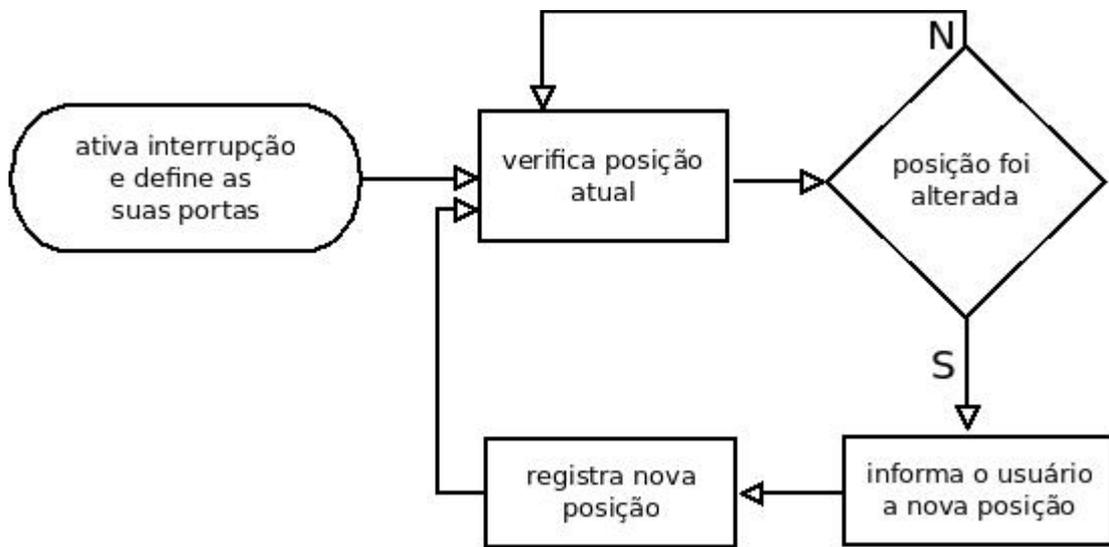


Figura 19: Fluxograma da programação do sensor direção do vento.

III.5. Sensor Termopar

Os termopares são compostos por dois metais dissimilares em contato, formando uma junção termoelétrica que produz uma voltagem proporcional à diferença de temperaturas entre as junções. Este fenômeno é conhecido como efeito Seebeck.

De acordo com o que Seebeck constatou em 1821, num circuito fechado, feito com fios de metais heterogêneo, irá fluir uma corrente elétrica proporcional à diferença de temperaturas entre uma junção T_1 e a outra extremidade T_2 , conforme ilustrado na figura 20.

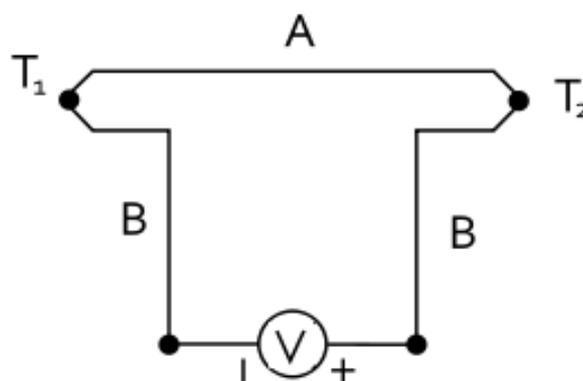


Figura 20: circuito utilizado por Seebeck.

Considerando esta hipótese verificou-se que a diferença de potencial (FEM) produzida é determinada pela expressão:

$$V = \int_{T_1}^{T_2} (S_B(T) - S_A(T)) dT \quad (3)$$

onde S_A e S_B são os coeficientes de Seebeck dos metais envolvidos na junção e T_2 e T_1 as temperaturas das duas junções.

Com base na utilização e no conhecimento mais comuns nos dias de hoje, segundo Thomazini e Albuquerque (2011), existe oito tipos de termoelementos padronizados que são: S, R, B, J, K, N, T e E. Eles cobrem uma faixa extensa de temperatura que vai de -200 a 2.300°C , conforme mostrado na figura 21.

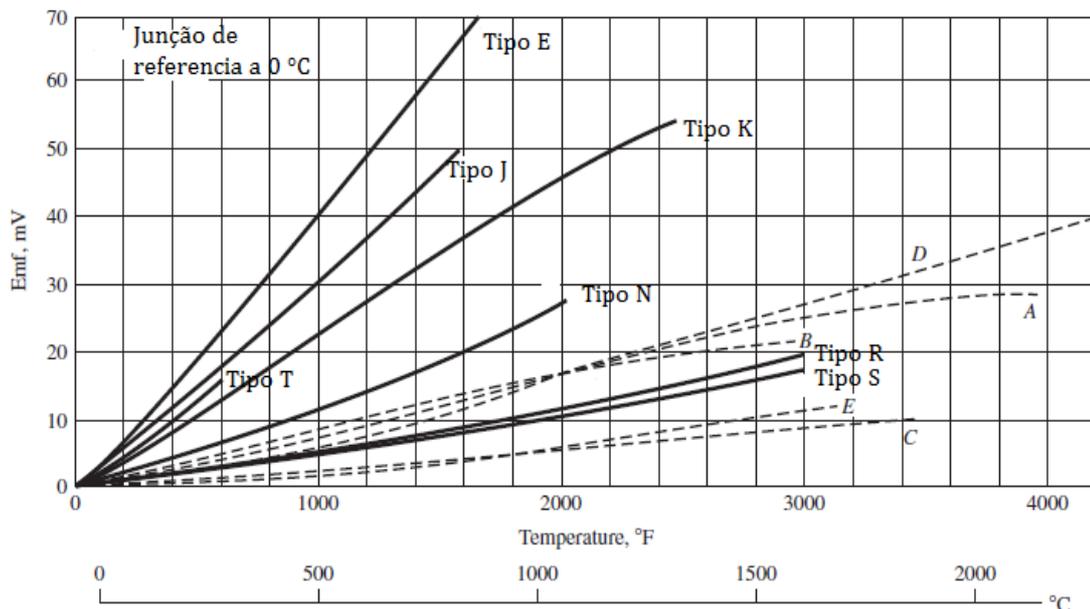


Figura 21: curvas de temperatura x milivoltagem.
(Adaptado de: Holman, J. P., 2012)

Com o intuito de aumentar o valor do potencial FEM a ser lido pelos sensores termopares, é possível combinar N pares de junções. Quando isto é feito, o resultando em voltagem é N vezes maior que o sinal de um único par, conforme mostrado na figura 22 onde o sinal foi multiplicado por 3.

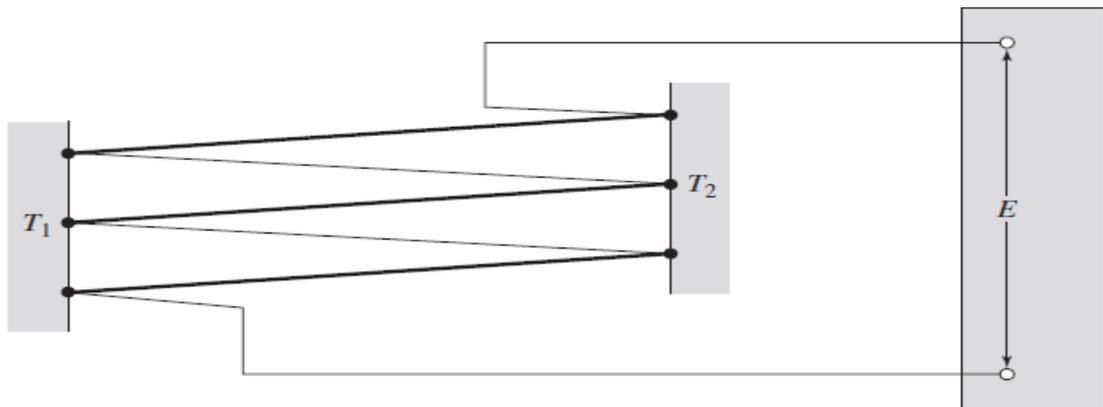


Figura 22: montagem de termopares com 3 junções para medição da temperatura T com base na temperatura de referência T_{ref} . (Holman, J. P., 2012)

Como citado, os termopares são uma combinação de diferentes materiais que geram uma diferença de potencial proporcional à temperatura. Entretanto, essa diferença de potencial é muito baixa para ser medida diretamente pelo módulo de leitura, por exemplo, o termopar tipo T fornece 0,00079 V à temperatura de 20 °C, com a junção de referência a 0°C, conforme equacionamento 04.

$$T = \sum_{i=0}^9 c_i V^i = c_0 + c_1 V + c_2 V^2 + c_3 V^3 + c_4 V^4 + c_5 V^5 + c_6 V^6 + c_7 V^7 + c_8 V^8 + c_9 V^9 \quad (4)$$

onde T é a temperatura da junção em °C, V^i é a medição da voltagem termoelétrica com a junção de referência a 0 °C e c_i são os coeficientes polinomiais para combinações de termopares, os quais são apresentados na tabela 2.

Para o uso dos sensores tipo termopares, é necessária a amplificação do sinal através de amplificadores operacionais. Nesse caso foi utilizado um circuito eletrônico, permitindo a aquisição dos dados analógicos e a sua conversão em digital, este circuito é dotado do chip ADS1115 da fabricante TEXAS INSTRUMENTS, que realiza a conversão do sinal de analógico para digital (ADC) com 16 bits de resolução, taxa de amostragem de 860 amostras por segundos e usa a transferência de dados via endereçamento I2C, permitindo com isso a ligação de 4 módulos com 4 sensores, totalizando a possibilidade de uso de 16 sensores termopares ou outro sensor o qual necessita a aquisição e conversão dos dados em digitais. Além disto, este sistema já possui um termistor para geração do sinal de 0°C como referência.

Tabela 2: Coeficientes polinomiais para a Eq. 4 para várias combinações de termopares padrão. (Adaptado de: Holman, J. P.,2012)

	Tipo E	Tipo J	Tipo K	Tipo R	Tipo S	Tipo T
	Cromel(+) vs. Constantan(-)	Ferro(+) vs. Constantan(-)	Cromel(+) vs. Alumel-5%(-) (Aluminum Silicon)	Platina-13% Ródio(+).vs. Platina(-)	Platina-10% Ródio(+).vs. Platina(-)	Cobre(+) vs. Constantan(-)
	-100 ° C to 1000 ° C ±0.5 ° C 9th Ordem	0 ° C to 760 ° C ±0.1 ° C 5th Ordem	0 ° C to 1370 ° C ±0.7 ° C 8th Ordem	0 ° C to 1000 ° C ±0.5 ° C 8th Ordem	0 ° C to 1750 ° C ±1 ° C 9th Ordem	-160 ° C to 400 ° C ±0.5 ° C 7th Ordem
C_0	1,04967E-01	-0,048868252	2,26585E-01	2,63633E-01	9,27763E-01	1,00861E-01
C_1	1,71895E+04	1,98731E+04	2,41521E+04	1,79075E+08	1,69527E+05	2,57279E+04
C_2	-282639,0850	-218614,5353	6,72334E+04	-48840341,37	-31568363,94	-767345,8295
C_3	1,26953E+07	1,15692E+07	2,21034E+09	1,90002E+10	8,99073E+09	7,80256E+07
C_4	-448703084,6	-264917531,4	-860963914,9	-4,82704E + 12	-1,63565E + 12	-9247486589
C_5	1,10866E+10	2,01844E+09	4,83506E+10	7,62091E+14	1,88027E+14	6,97688E+11
C_6	-1,76807E + 11		-1,18452E + 12	-7,20026E + 16	-1,37241E + 1	-2,66192E + 13
C_7	1,71842E+12		1,38690E+13	3,71496E+18	6,17501E+17	3,94078E+14
C_8	-9,19278E + 12		-6,33708E + 13	-8,03104E + 19	-1,56105E + 19	
C_9	2,06132E+13				1,69535E+20	

A comunicação entre o módulo e a placa é realizada pelo protocolo I2C, deve-se fazer a ligação do módulo ADC ao Arduino UNO através dos pinos A4 e A5, pré-definidos para esta finalidade, conforme mostrado na figura 23.

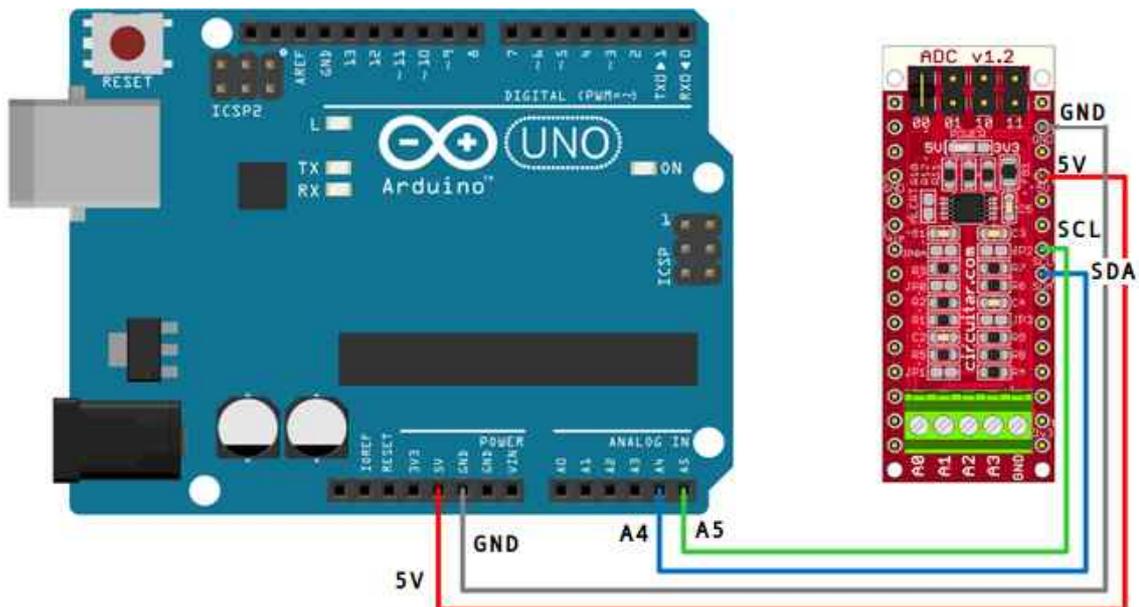


Figura 23: Croqui da ligação elétrica do módulo ADC ao arduino UNO.

III.5.1. Programação sensor termopar para medidas de diferenças de temperaturas

Como o circuito ADC trabalha com comunicação I2C, é necessário iniciar a programação no Arduino incluindo a biblioteca que irá fazer essa comunicação, através da linha de comando `#include <Wire.h>` e dentro da estrutura `void setup()`, deve-se iniciá-la com a linha de comando `Wire.begin()`.

Outra biblioteca que deve ser incluída é a `Nanoshield_ADC`, a qual recebe o sinal de acordo com o endereçamento, fornece o ganho programado e processa o sinal, comparando o seu valor de voltagem com o GND do arduino. Este esquema consegue obter uma boa precisão de medida mesmo em condições onde existem baixas tensões diferenciais entre dois terminais. Um exemplo da programação com o ajuste do amplificador interno para ganho de 1 na resolução (sem amplificação) com a leitura de um termopar tipo S no canal 0 e o mapeamento da voltagem do módulo no canal 1 pode ser visto a seguir. Há a opção de ganho de 2/3, 1, 2, 4, 8 e 16, porem o fundo de escala tem a queda de 6,144V para 0,256V.

```
#include <Wire.h>
#include <Nanoshield_ADC.h>

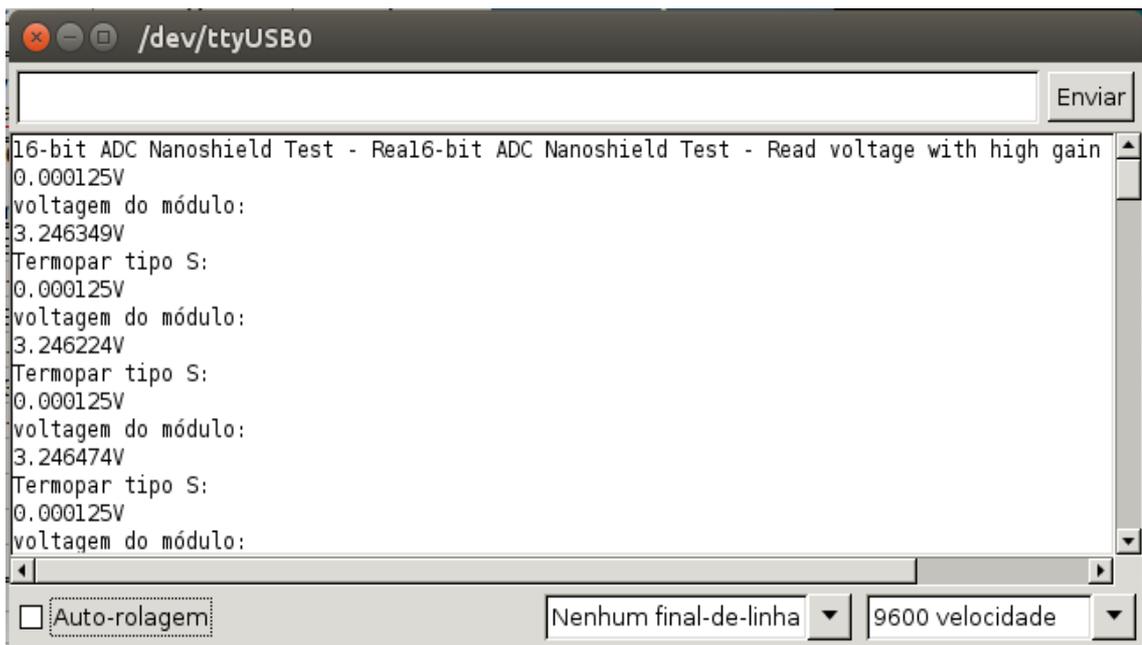
Nanoshield_ADC adc;
int channel = 0;

void setup()
{
  Serial.begin(9600);
  Serial.print("16-bit ADC Nanoshield Test Read voltage with high gain");
  adc.begin(); //ativa o módulo

  // Possibilidades de ajuste do amplificador
  // GAIN_TWOTHIRDS = 6.144V para o fundo de escala
  // GAIN_ONE = 4.096V para o fundo de escala
  // GAIN_TWO = 2.048V para o fundo de escala
  // GAIN_FOUR = 1.024V para o fundo de escala
  // GAIN_EIGHT = 0.512V para o fundo de escala
  // GAIN_SIXTEEN = 0.256V para o fundo de escala
  adc.setGain(GAIN_ONE); //define o ganho na leitura
}

void loop()
{
  Serial.println("Termopar tipo S: ");
  Serial.print(adc.readVoltage(channel), 6);
  Serial.println("V");
  Serial.println("voltagem do módulo: ");
  Serial.print(adc.readVoltage(1), 6);
  Serial.println("V");
  delay(1000);
}
```

O resultado do teste da programação do conversor analógico digital é mostrado na figura 24, onde está sendo mapeado no canal 1 a voltagem de saída do módulo de 3,3 V e no canal 0 do conversor está sendo feita a leitura da voltagem do termopar tipo S que está na temperatura ambiente de 25 °C. O inconveniente deste esquema de leitura reside no fato que ele não possui o ponto de gelo de referência.



The image shows a terminal window titled "/dev/ttyUSB0" with a "Enviar" button. The terminal output displays the following text:

```
16-bit ADC Nanoshield Test - Real16-bit ADC Nanoshield Test - Read voltage with high gain
0.000125V
voltagem do módulo:
3.246349V
Termopar tipo S:
0.000125V
voltagem do módulo:
3.246224V
Termopar tipo S:
0.000125V
voltagem do módulo:
3.246474V
Termopar tipo S:
0.000125V
voltagem do módulo:
```

At the bottom of the terminal window, there are controls for "Auto-rolagem" (unchecked), "Nenhum final-de-linha" (selected), and "9600 velocidade" (selected).

Figura 24: Teste do conversor analógico digital.

Uma segunda alternativa para uso de termopares com Arduino é o circuito interno fabricado pela MAXIM, identificado como MAX6675, figura 25, o qual possibilita a conversão do sinal analógico do termopar em digital e apresentação do valor na escala de temperatura Celsius ou Fahrenheit, com resolução de 12 bit e apresentação do valor em 0,25°C. Neste circuito há a compensação da junção “fria”, não necessitando a junção do segundo ponto do termopar, porem o circuito MAX6675 foi projetado para trabalhar somente com o termopar Tipo K e 1 termopar por circuito, se houver a necessidade de trabalhar com diversos termopares, deve-se usar um circuito para cada sensor .

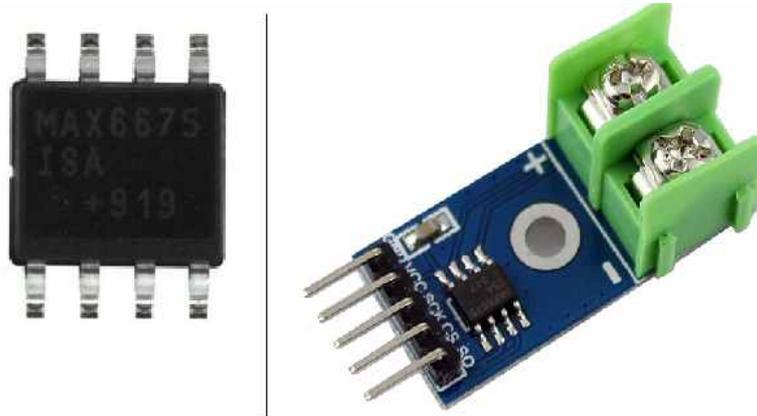


Figura 25: CI MAX6675 a esquerda na imagem e sua montagem em uma placa de fenolite com os bornes de conexão e capacitor de 0,1µF.

A programação para uso do CI6675 é simplificada se usado a biblioteca *max6675.h*. Para obter o valor da temperatura do termopar tipo K deve-se usar o comando *thermocouple.readCelsius()*, conforme mostrado na programação abaixo.

```
#include "max6675.h"//biblioeca do CI Max6675

//define as portas do MAX6675
int thermoS0 = 4;//pino 4 - sinal
int thermoCS = 5;//pino 5 - seleção do CI
int thermoSCK = 6; //clock
MAX6675 thermocouple(thermoSCK, thermoCS, thermoS0);

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  delay(300);//tempo para estabilidade do sensor
  Serial.print (F("temperatura MAX6675: "));
  Serial.print (thermocouple.readCelsius());//obtem a temperatura
  Serial.println (" C");
}
```

III.6. Módulo de Calendário e Hora

O módulo DS1307, fabricado pela *Maxim Integrated Products*, mostrado na figura 26, fornece para o Arduino o calendário e a hora atual na formatação de segundos, minutos, horas, dia, data, mês e ano. O seu consumo de energia é baixo e possui bateria externa para fornecer a energia necessária para sua operação caso a energia provinda do Arduino seja interrompida, mantendo assim o calendário e a hora atualizada.

A comunicação deste equipamento é realizado com o protocolo I2C e pode trabalhar em paralelo com outros dispositivos que use a comunicação serial I2C, semelhante ao conversor analógico digital citado neste trabalho. Há a possibilidade de acoplar nesse módulo o sensor de temperatura DS18B20, simplificando assim o uso deste sensor, pois a energia será fornecida pelo módulo.

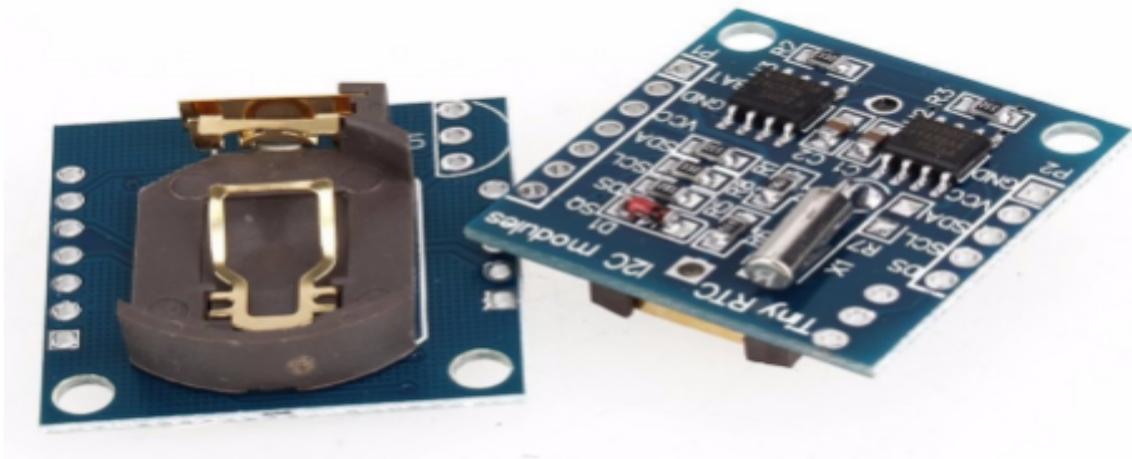


Figura 26: Módulo relógio e calendário DS1307.

Para uso do módulo relógio e calendário é necessário o uso de duas bibliotecas, a *wire.h* que será responsável pela comunicação I2C disponível em <https://www.arduino.cc/en/Main/Software> e a *DS1307.h*, que irá ajustar a hora e o calendário além de obter e registrar outras informações necessárias.

A conexão do módulo DS1307 com o arduino UNO ocorre pelas porta A4 e A5 com os pinos SDA e SCL do módulo, pois é onde encontra-se configurado a comunicação I2C com os periféricos; No caso do módulo discutido, ele pode ser usado em outras portas se alterado o

endereço de comunicação na biblioteca wire.h e DS1307.h. Apesar disto, esta não é uma alternativa recomendável, pois pode interferir na comunicação com outros periféricos.

Para uso deste dispositivo deve-se inserir também uma bateria que manterá a sua alimentação externa de forma a manter a data e hora local pré-programados. Esse ajuste no calendário será realizado uma única vez enquanto houver energia na bateria, conforme mostrado na programação que segue sendo as definições de data, hora, mês e ano estabelecidas nas linhas 12 a 15.

```
#include <DS1307.h> //carrega a biblioteca
1.DS1307 rtc(SDA, SCL); //instancia o objeto rtc, associando as informações da
2.                               // biblioteca e informa as portas usadas
3.
4.void setup()
5.{
6.Serial.begin(115200); // ajusta a velocidade de comunicação
7.rtc.begin(); // inicia o objeto rtc
8.
9.// Set the clock to run-mode
10.rtc.halt(false); // set o relógio no modo run-mode
11.
12.// comente ou apague as linhas após ajuste da data e hora
13.rtc.setDOW(FRIDAY); // ajusta o dia como sendo sexta-feira
14.rtc.setTime(12, 0, 0); // ajusta o tempo em 12:00:00 (formato 24hr)
15.rtc.setDate(4, 1, 2017); // ajusta o calendário em 1/04/2017
16.}
17.
18.void loop()
19.{
20.// mostra o dia da semana
21.Serial.print(rtc.getDOWStr());
22.Serial.print(" ");
23.
24.// mostra a data
25.Serial.print(rtc.getDateStr());
26.Serial.print(" -- ");
27.
28.// mostra a hora
29.Serial.println(rtc.getTimeStr());
30.
31.// espera 1 segundo e inicia a rotina
32.delay (1000);
33.}
```

III.7. Sensor de Temperatura DS18B20

O sensor DS18B20 é um sensor de temperatura digital fabricado pela Maxim Integrated Products, com medição de temperatura na faixa de -55°C a 125°C com acuracidade de $\pm 0,5^{\circ}\text{C}$ na faixa de temperatura de -10°C a 85°C e resolução programável de 9 bits a 12 bits. Este sensor tem a característica de se conectar com outro dispositivo através da comunicação *onewire*, conforme ilustrado na figura 27, o que significa que diversos dados poderão ser transmitidos por um único conector, identificado pela sigla DQ. Os outros dois conectores do sensor são destinados à alimentação, sendo estes o ground (GND) e 5V (V_{CC}). Cabe destacar entretanto que para a utilização deste método de comunicação, conforme orientação do próprio fabricante, é recomendável a ligação de um resistor de $4,7\text{ K}\Omega$ entre a fonte de alimentação 5V e o pino DQ. Apesar da comunicação ser feita via um conector (DQ), é possível conectar um grande número de sensores DS18B20 a este condutor, pois cada sensor possui um código único de 64 bits que permite identificar de qual deles o sinal provém.

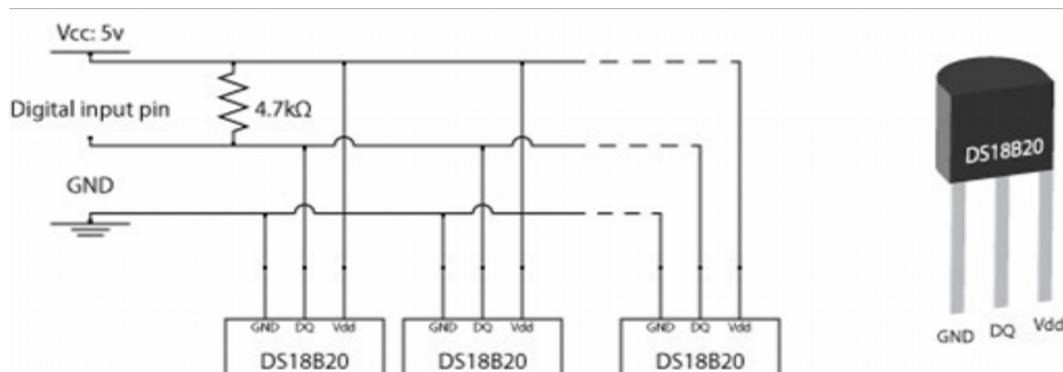


Figura 27: Ligação de múltiplos sensores de temperatura DS18b20.

Para o uso deste sensor com o Arduino é necessário o uso de duas bibliotecas específicas na programação, a *DallasTemperature* e a *Onewire*, ambas disponíveis em <https://www.hacktronics.com/Tutorials/arduino-1-wire-address-finder.html>. A primeira biblioteca é responsável pela obtenção e conversão dos dados, identificação do código e endereçamento de cada sensor, a segunda é usada para fazer a comunicação *Onewire* entre os dispositivos.

Caso haja a necessidade de usar vários sensores, conforme ilustrado na figura 27, deve-se obter o código de identificação de cada sensor, com o auxílio das bibliotecas citadas e

comandos específicos para essa função. Um esquema fácil de identificação que tem sido usado com sucesso no laboratório é apresentado na sequência. Para implementar essa programação na placa Arduino é necessário que os sensores estejam conectados, pois serão identificados a quantidade de sensores e o código de cada sensor. Para isto, o código apresenta o endereçamento do sensor que apresente variação de temperatura maior que 1,5°C, permitindo a sua identificação física associado ao respectivo endereço. A lógica de operação desta programação está ilustrada no fluxograma apresentado na figura 28 e o seu código apresentado na sequência:

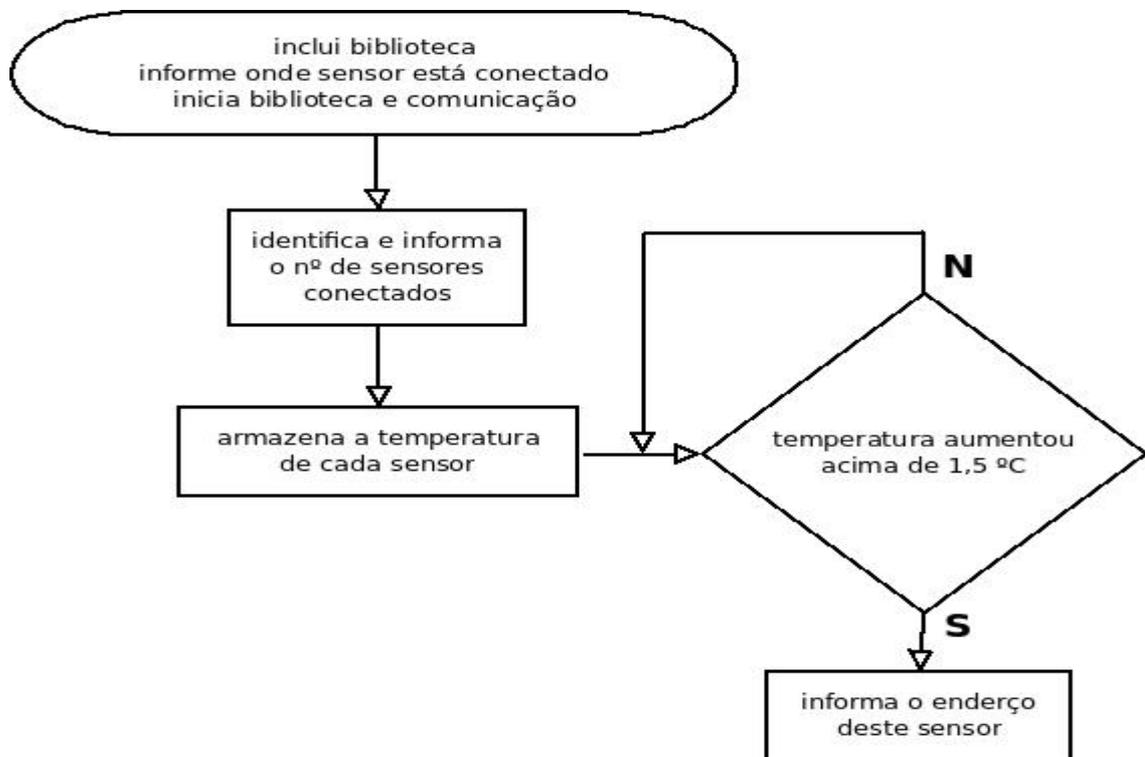


Figura 28: Lógica de operação da programação que identifica o endereço do sensor Dallas DS18B20

```

#include <OneWire.h>
#include <DallasTemperature.h>

// Sensor de temperatura conectado na porta digital 2
#define ONE_WIRE_BUS 2
//Resolução da temperatura 9 a 12 - 9=0.5; 10=0.25; 11=0.125; 12=0.0625
#define TEMPERATURE_PRECISION 10
//Setup a oneWire para comunicação com qualquer dispositivo Onewire
OneWire oneWire(ONE_WIRE_BUS);
  
```

```

//Passa a referencia Onewire para o sensor de temperatura
DallasTemperature sensors(&oneWire);

//vetor para armazenar o endereço dos sensores
DeviceAddress sensor;

int nsen;
int i=0;
float T0[100]; //vetor para armazenar a temperatura inicial
float T[100]; //vetor para armazenar a temperatura posterior
boolean inicia=true;

void setup()
{
  Serial.begin(9600); //inicia comunicação serial
  sensors.begin(); // inicia a biblioteca do sensor
  nsen=sensors.getDeviceCount(); //computa o numero de sensores conectados
  Serial.print("Foram encontrados ");
  Serial.print(nsen);
  Serial.println(" sensores");
}

void printAddress(DeviceAddress deviceAddress)
{
  for (uint8_t i = 0; i < 8; i++)
  {
    Serial.print("0x");
    if (deviceAddress[i] < 16) Serial.print("0");
    Serial.print(deviceAddress[i], HEX);
    if (i<7) Serial.print(", ");
    else Serial.println(" };");
  }
}

void loop()
{
  sensors.requestTemperatures(); //envia comando para obter as temperaturas
  if(i<nsen) //i muda quando reconhece um sensor
  {
    if(inicia)
    {
      int j=0;
      for (j=0;j<nsen;j++) //j muda depois de avaliar a temperatura
      {
        sensors.getAddress(sensor,j);
        sensors.setResolution(sensor, TEMPERATURE_PRECISION);
        T0[j]=sensors.getTempC(sensor);
      }
    }
  }
}

```

```

    }
    inicia=false;
    Serial.print("//Endereco do Sensor");
    Serial.println(i+1);
  }
  delay(200);
  int k;
  for (k=0;k<nсен;k++)
  {
    sensors.getAddress(sensor,k);
    T[k]=sensors.getTempC(sensor);
    if (T[k]-T0[k]>1.5)
    {
      Serial.print("DeviceAddress Sensor");
      Serial.print(i+1);
      Serial.print(" = { ");
      printAddress(sensor);
      i++;
      inicia=true;
      delay(5000);
      break;
    }
  }
}
}
}

```

Com a identificação física e numérica (endereçamento) dos sensores, pode-se elaborar um código capaz de ler a temperatura de cada sensor. Para isto é necessário o uso das bibliotecas de comunicação do sensor e o comando *getTempC(deviceAddress)*, conforme ilustrado na programação a seguir:

```

#include <OneWire.h>
#include <DallasTemperature.h>

OneWire oneWire(2); //sensores conectados na porta 2-necessita resistor 4,7kΩ
DallasTemperature sensors(&oneWire);
//declara o endereço dos sensores 1,2 e 3, obtido programação anterior
DeviceAddress tempSensor1 = {0x28, 0xFF, 0x2B, 0x45, 0x4C, 0x04, 0x00, 0x10};
DeviceAddress tempSensor2 = {0x28, 0x7C, 0x8A, 0x5D, 0x05, 0x00, 0x00, 0xFD};
DeviceAddress tempSensor3 = {0x28, 0xE5, 0x82, 0x9D, 0x04, 0x00, 0x00, 0x76};

float tempC; //variável que armazena temporariamente a temperatura
int i=0; //variável que armazena em qual leitura está, iniciando do zero

void setup(void)
{

```

```

Serial.begin(9600);
//Ajusta a resolução – opção de 9, 10, ou 11 bit.
sensors.setResolution(tempSensor1, 10);//resolução sensor 1
sensors.setResolution(tempSensor2, 10);//resolução sensor 2
sensors.setResolution(tempSensor3, 10);//resolução sensor 3
}

void loop(void)
{
/*****Rotina do sensor 1*****/
sensors.requestTemperaturesByAddress(tempSensor1);/*comando para obter a
                                                    temperatura do sensor 1*/
tempC = sensors.getTempC(tempSensor1);//registra a temperatura na variável
Serial.println(" ");
Serial.print("leitura ");
Serial.println(i);
Serial.print("Temperatura sensor 1: ");
Serial.print(tempC,4); //imprime a temperatura do sensor 1
Serial.print(" C: ");
Serial.print(" - ");
Serial.print(DallasTemperature::toFahrenheit(tempC),4);/*converte
temperatura de graus Celsius para
Fahrenheit*/
Serial.println(" F");

/*****Rotina do sensor 2*****/
/*comando para obter a temperatura do sensor 1*/
tempC = sensors.getTempC(tempSensor2);//registra a temperatura na variável
Serial.println(" ");
Serial.print("leitura ");
Serial.println(i);
Serial.print("Temperatura sensor 2: ");
Serial.print(tempC,4); //imprime a temperatura do sensor 1
Serial.print(" C: ");
Serial.print(" - ");
Serial.print(DallasTemperature::toFahrenheit(tempC),4);/*converte
temperatura de graus Celsius para
Fahrenheit*/
Serial.println(" F");

/*****Rotina do sensor 3*****/
/*comando para obter a temperatura do sensor 1*/
tempC = sensors.getTempC(tempSensor3);//registra a temperatura na variável
Serial.println(" ");
Serial.print("leitura ");
Serial.println(i);
Serial.print("Temperatura sensor 3: ");
Serial.print(tempC,4); //imprime a temperatura do sensor 1
Serial.print(" C: ");

```

```
    Serial.print(" - ");
    Serial.print(DallasTemperature::toFahrenheit(tempC),4);/*converte
temperatura                               de graus Celsius para
Fahrenheit/*
    Serial.println(" F");

    delay(1000);
    i++;//soma 1 para numerar próxima tomada de temperaturas.
}
```

Essa programação usa o código de identificação de cada um dos 3 sensores para obter a temperatura deles em graus Celsius e faz a conversão em graus Fahrenheit. O valor da temperatura foi ajustado para ter uma resolução de 10 bits e impressão com 4 casas decimais.

IV - RESULTADOS

Após estudo, análise e definição dos sensores mais relevantes para pesquisa envolvendo coletores solares e as possibilidades de integrá-los à placa Arduino, foi elaborada a programação de cada sensor. Posteriormente esta programação foi reformulada para otimização e transformação em biblioteca, facilitando o seu uso e assegurando a integridade da programação original.

Todos os sensores e dispositivos foram integrados a placa Arduino UNO para formar um módulo de coleta de dados. Para isso, foi construído um circuito eletrônico composto por bornes para conexões extras, por resistores necessários aos sensores, pelo leitor de cartão SD e por pinos extras de conexão. A partir desta montagem, levantamentos de alguns dados foram realizados assegurando a funcionalidade do sistema.

IV.1. Elaboração de Bibliotecas

Bibliotecas são subprogramas que auxiliam o programa principal através da execução de funções e compartilhamento de dados. O uso desta alternativa de montagem do código o torna modular, facilitando ao usuário e ao programador manipular e montar códigos adequados para alcançar o objetivo desejado.

Uma biblioteca possui dois arquivos obrigatórios, sendo eles o arquivo cabeçalho e o arquivo com o código fonte. O cabeçalho possui basicamente uma listagem de tudo que há na biblioteca, como funções e variáveis, sendo identificado como um arquivo de extensão .h. O

arquivo com o código fonte possui o código que executa as funções, semelhante ao programa principal e é identificado como um arquivo de extensão .cpp.

Antes de iniciar a elaboração de uma biblioteca é importante listar tudo o que terá essa biblioteca, analisar a finalidade de cada variável e função, com o objetivo de otimizá-las. Neste processo, se possível, se refaz a programação de forma a eliminar variáveis desnecessárias e tornar o *sketch* do programa menor e mais eficiente. Tomando como exemplo de reformulação para o uso de bibliotecas será avaliado o programa dedicado ao sensor de vazão, apresentado no item III.2.1 deste trabalho. Neste código aparecem algumas variáveis, a declaração onde o sensor foi conectado e duas funções, a *rpm* e *calc*. Para essa biblioteca ser flexível e permitir seu uso em diversos sensores similares concebeu-se uma função que calcula o valor medido a partir de uma equação de primeiro grau do tipo $f(x)=ax+b$. Nesta biblioteca cabe ao usuário informar em qual porta conectou o sensor, o valor das variáveis *a* e *b* da função linear, previamente obtidas na calibração do sensor, o número de pulsos que se pretende medir e o tempo limite para espera destes pulsos. Este último parâmetro é de fundamental importância pois permite abortar a função quando houver algum erro na leitura e desconsiderar estes dados. Caso haja novas tarefas a serem executadas além desta medida, a execução do código continua.

Deve-se observar que cada modelo de placa de processamento do Arduino tem os locais próprios para conectar esses sensores (vazão e anemômetro), como por exemplo a placa UNO tem os pinos 2 e 3 específicos para essa função. No Arduino MEGA os pinos são 2, 3, 18, 19, 20 e 21 podem desempenhar a mesma função. Ainda é possível definir a função de interrupção a outros pinos no Arduino UNO fazendo uso de comandos e bibliotecas adicionais.

Com o uso da biblioteca, a programação ficou resumida em algumas linhas, para o sensor de vazão, ficou composta pelos comandos para calcular a vazão, *vazao.calcvazao*, e a impressão na serial do seu valor, *Serial.print(vazao.vazao)*. Caso seja necessário, está também disponível na biblioteca *Vazao.h* o comando para mostrar o tempo médio entre os pulsos, o qual pode ser usado para auxiliar na calibração do equipamento. Este tempo é armazenado na variável *vazao.Totaltime*, após execução do comando *vazao.calcvazao*.

A seguir é mostrado o exemplo da programação otimizada do sensor de vazão, com o uso da biblioteca vazão.h.

```
#include <Vazao.h> // inclui a biblioteca vazão
Vazao vazao(7.27,3.645,5,5000,2); /* vazao(calibracao a,calibracao b,
numero pulsos para calculo médio,tempo limite(em ms)para calculo da
media,porta conectada ao sensor)- equacao f(x)= a.x+b */

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  attachInterrupt(digitalPinToInterrupt(vazao.porta),contapulsovazao,RISING);
/*ativa a função de interrupção da porta 2 ou 3 para UNO e 2,3,18,19,20,21 para
MEGA, executa a função contapulso quando o sinal vai de baixo para alto */

  vazao.calcvazao(); //função que calcula a vazão conforme dados informado
  Serial.print("vazao: ");
  Serial.print(vazao.vazao);
  Serial.println(" l/h");
  //Serial.print(vazao.Totaltime);/*caso necessite do tempo médio dos pulsos,
  retire as duas barras do início do paragrafo */
  delay (1000);
}

void contapulsovazao () /*função que conta os pulsos do sensor hall no tempo
informado*/
{ vazao.contador++;}
```

Para usar a função interrupção em outras portas no Arduino UNO, além da 2 e 3 que estão pré-programadas, é necessário ativá-las com alguns comandos específicos. Para este fim é utilizada a biblioteca *PinChangeInt.h*, que traz os comandos necessários para ativar a interrupção dos pinos desejados, ela está disponível em:

<http://playground.arduino.cc/Main/PinChangeInt>

Segue um exemplo de programação, com comentários explicativos de cada linha, para o sensor de vazão, usando o pino 8 do Arduino UNO. Inicialmente a biblioteca *PinChangeInt.h* ativa a interrupção do *Port B, C e D* (portas 8 a 13, A0 a A5 e 0 a 7), deve-se desabilitar a função interrupção nas portas que não serão usadas para economia de memória, neste caso *Port C e D*, mantendo somente as portas 8 a 13.

```

#include <Vazao.h> //inclui a biblioteca vazão
#include <PinChangeInt.h> // biblioteca de interrupção para outras portas
#define NO_PORTC_PINCHANGES //desabilita interrupção port C
#define NO_PORTD_PINCHANGES //desabilita interrupção port D

Vazao vazao(7.27,3.645,5,5000,8); // vazao(calibracao a,calibracao b,
//numero pulsos para calculo medio,tempo limite(em ms)para calculo da
//media,porta conectada ao sensor)- equacao a.x+b

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  PCintPort::attachInterrupt(vazao.porta,contapulsovazao, RISING);
  /*ativa a função de interrupção da porta 8, executa a função contapulso
quando o sinal vai de baixo para alto*/
  vazao.calcvazao(); //função que calcula a vazão conforme dados informado
  Serial.print("vazao: ");
  Serial.print(vazao.vazao);
  Serial.println(" l/h");
  delay (1000);
}

void contapulsovazao ()//função que conta os pulsos do sensor hall
{ vazao.contador++; }

```

A biblioteca usada para o anemômetro é semelhante à biblioteca para o sensor de vazão, pois ambos os aparelhos usam no seu circuito eletrônico o sensor tipo Hall. A curva característica de calibração deste equipamento também é do tipo linear, $f(x) = ax + b$, diferenciando somente nas constantes a e b , obtidos na sua calibração. Para uso da biblioteca, deve-se iniciar a programação com a inclusão da biblioteca *Anemometro.h* e em outra linha, informar os dados do equipamento e da medição. No *loop* da programação ativa-se a interrupção com o comando *attachInterrupt*, calcula-se a velocidade do vento com o comando *anemometro.calcvelocidade()* e imprimir na tela do computador o valor calculado com o comando *Serial.print (anemometro.velocidade)*, conforme demonstrado na programação a seguir.

```

#include <Anemometro.h>

Anemometro anemometro(1.6736,0.9984,3,4000);

```

```

/* anemômetro (calibração a,calibração b,numero pulsos em uma volta para
calculo médio,tempo limite (em ms)para calculo da média)- equacao F(x)= a.x+b
int porta;//anemometro porta=3,sensor vazão porta=2(opcional qual porta usar).
void setup()
{ Serial.begin(9600); }
void loop()
{
  porta=3;
  attachInterrupt(digitalPinToInterrupt(anemometro.porta),contapulso, RISING);
  anemometro.calcvelocidade();
  Serial.print ("velocidade media do vento: ");
  Serial.print (anemometro.velocidade);
  Serial.println (" m/s");*/
  delay (40000);
}
void contapulso () //função que conta os pulsos do sensor hall
{
  //if (porta ==2) {vazao.contador++;} //se estiver usando sensor de vazão
  //retire a barra dupla antes do if.
  if (porta ==3) {anemometro.contador++;} //se estiver usando anemômetro
  //retire a barra dupla antes do if.
}

```

IV.2. Calibração do anemômetro de caneca

Para o anemômetro abordado neste trabalho, o qual foi confeccionado em plástico PLA e composto pelo sensor de efeito Hall, gerando 3 pulsos por revolução, foi realizada a calibração e definida a equação da velocidade. Para isso foi usado um sistema de ventilação, com controle de velocidade obtido pelo Dimmer Bivolts de 600 W e um anemômetro de fio quente, fabricante OMEGA, modelo HHF-SD1, capacidade de leitura de 0,7 a 72 km/h, resolução de 0,01 km/h no intervalo até 18 km/h e exatidão na leitura de 5% + 0,3 km/h.

A calibração ocorreu na faixa de velocidade de 3,5 a 9,5 km/h, com a coleta de 56 dados de frequência dos pulsos, possibilitando a geração dos gráficos com as linhas de

tendência da equação linear, exponencial, de segundo e terceiro grau, ambas mostradas na figura 29.

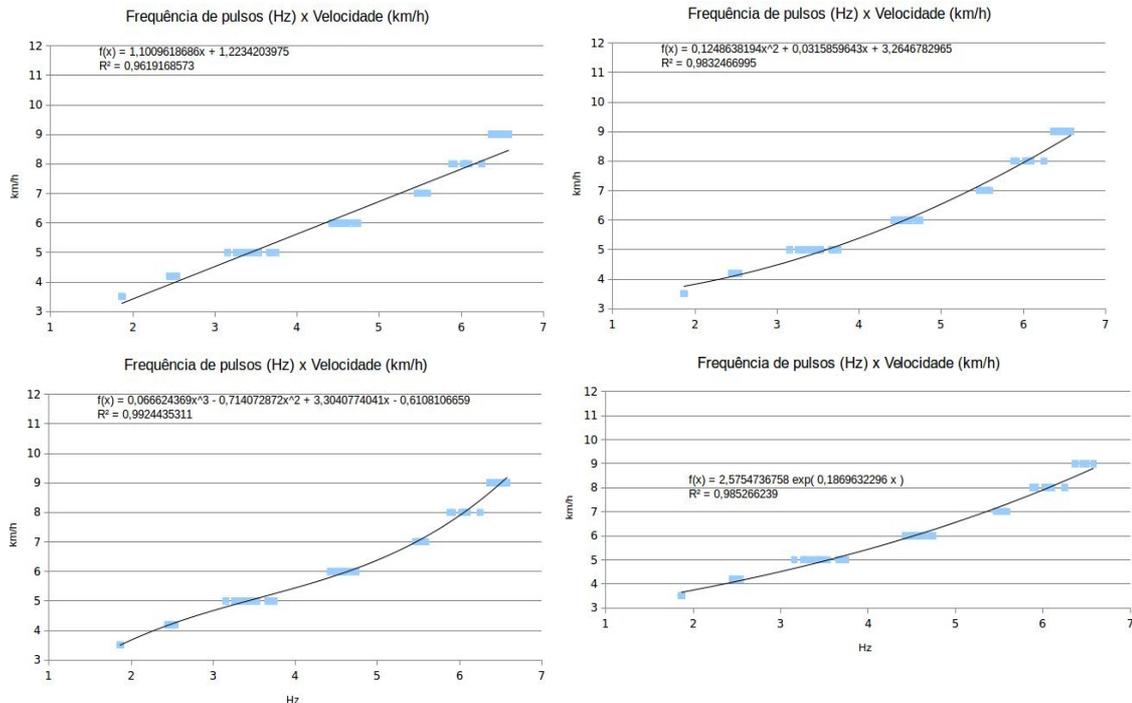


Figura 29: Curvas de calibração do anemômetro de caneca com linha de tendencia com equações linear, de segundo e terceiro grau e exponencial.

De acordo com Venkateshan (2015), o desvio padrão é obtido pela equação 5 e a média do valor da amostragem pela equação 6.

$$\sigma_e^2 = \frac{\sum_{i=1}^n (x_i - m_s)^2}{n-1} \quad (5)$$

$$m_s = \sum_{i=1}^n \left(\frac{x_i}{n} \right) \quad (6)$$

onde x_i é o valor da amostra i , m_s é a média da amostragem, σ_e é o desvio padrão e n o numero de amostragem.

Com base nas equações das linhas de tendências apresentadas na figura 29, realizou-se para cada uma delas o cálculo da variância amostral, do desvio padrão e a tolerância com

intervalo de confiança de 95% para a distribuição Student com amostragem $d = 50$, conforme apresentado na tabela 3.

Tabela 3: Valores estatísticos para amostragem com diferentes equações da linha de tendencia.

	variança amostral σ^2 (km/h) ²	desvio padrão σ (km/h)	Tolerância km/h
equação linear $V=1,10.x + 1,223$	0,096	0,310	0,622
equação grau 2 $V=0,125.x^2+0,031.x+3,265$	0,042	0,205	0,412
equação grau 3 $V=0,0666.x^3-0,714.x^2+3,304.x-0,6108$	0,019	0,138	0,277
equação exponencial $V=2,575.e^{(0,187.x)}$	0,040	0,200	0,403

Os dados estatísticos apresentados na tabela 3 mostram que a equação que melhor representa a velocidade do ar detectado pelo anemômetro de caneca é de grau três, com desvio padrão de 0,138 km/h e tolerância de $\pm 0,277$ km/h. Desta forma, caso seja necessária o uso da equação de grau 3, há uma biblioteca dedicada a essa aproximação. Entretanto, como a aproximação linear já fornece um resultado adequado ao que o trabalho se propõe, e é bem mais simples de ser obtida, ela será mantida como padrão na biblioteca.

IV.3. Montagem do módulo de coleta de dados

Para o uso do sensor de vazão e do anemômetro é necessário a placa Arduino, resistor de 10 k Ω e a programação específica dos sensores. Para uso de outro tipo de sensores, como termopares, termistor e semicondutores, são necessários componentes complementares a serem acoplados nesta placa. Para atender a essa necessidade, foi elaborado um módulo de coleta de dados com a possibilidade de uso dos sensores mais comuns em sistemas de aquisição de dados da área da termodinâmica e transferência de calor, sendo eles:

- Sensor de temperatura Termopares;
- Sensor de temperatura Termistor;
- Sensor de temperatura fabricante Dallas, modelo DS18B20;
- Sensores de vazão;
- Anemômetros;
- Radiômetros;

- Módulo Calendário e hora;
- Leitor de cartão SD;

O módulo possibilita ainda o registro dos dados dos sensores em um cartão SD, no formato texto sem formatação (extensão .txt), com a possibilidade de incluir a data e hora da coleta dos dados. A figura 30 mostra o módulo e a integração de alguns dispositivos a ele.

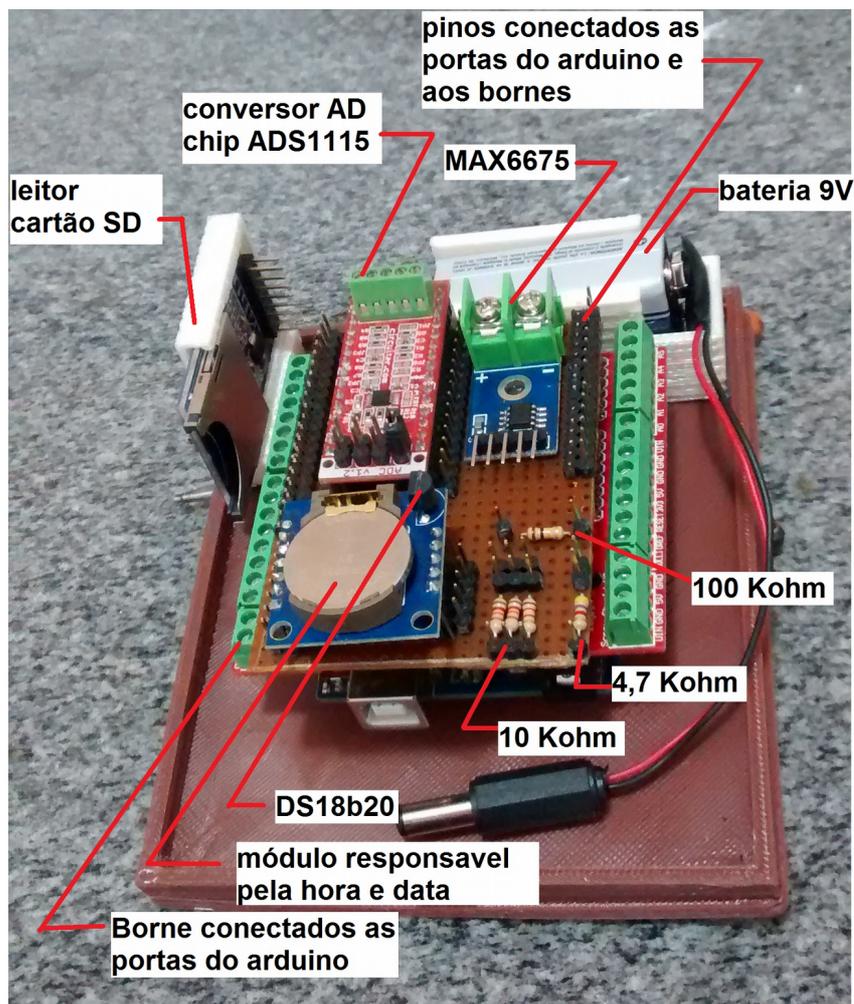


Figura 30: Módulo de coleta de dados.

Os principais componentes do módulo de coleta de dados são:

- Borne: permite a conexão de cabos, com ou sem terminais, nas portas do Arduino. Esses bornes também estão ligados aos pinos na placa superior,

permitindo interligar componentes eletrônicos, como resistor ou capacitor, sem a necessidade de alterar a ligação do componente que está conectado ao borne.

- Módulo de data e hora (RTC DS1307): com 56 bytes de memória não-volátil disponível para uso, pode armazenar e fornecer informações completas de data com o dia da semana, dia do mês, mês, ano, horas, minutos e segundos, nos formatos de 12 ou 24 horas. A comunicação é realizada pelo protocolo I2C, sendo necessário somente duas conexões para a comunicação. Permite a integração do sensor de temperatura DS18B20 fabricado pela Dallas Semiconductor, o que é útil para mapear a temperatura do módulo e evitar superaquecimento ou realizar uma medida de temperatura externa.
- Conversor Analógico Digital: composto pelo CI ADS1115, possibilitando leitura de sensores industriais de temperatura, pressão, umidade, entre outros, além de monitoramento de bateria, tensões de alimentação ou qualquer outro projeto que necessite de conversão analógico-digital com disponibilidade de 4 canais, 16 bits de resolução, amplificador interno e filtro RC.
- MAX6675: Desenvolvido para ligação de termopar do tipo K com compensação da junta fria, resolução de 0,25 °C, leitura até 1024 °C, voltagem de trabalho de -0,3 a 6 V, corrente de trabalho 50 mA e temperatura de operação do CI na faixa de -20 °C a 85 °C.
- Resistor 10 kΩ: Necessário quando se usa equipamentos implementados com sensores tipo Hall ou ReedSwitch.
- Resistor 4,7 kΩ: Usado com o sensor de temperatura DS 18B20, um único resistor necessário para uso de vários sensores.
- Resistor 100 kΩ: Necessário para uso de termistor de 100 kΩ, útil para mapear temperaturas até 260 °C.

A figura 31 mostra o módulo com os componentes ligados e coletando os dados, os quais estão sendo armazenados no cartão SD em um arquivo de texto identificado como DADOS.TXT. Os mesmos dados estão sendo mostrados na tela do computador simultaneamente, caso não haja necessidade do cartão SD. A programação foi elaborada de

forma a identificar a presença do cartão de forma a alertar, através de uma mensagem na tela do computador, sobre sua ausência. No Apêndice C há o esquema elétrico detalhado deste módulo e figuras da placa de circuito impresso para impressão e confecção.

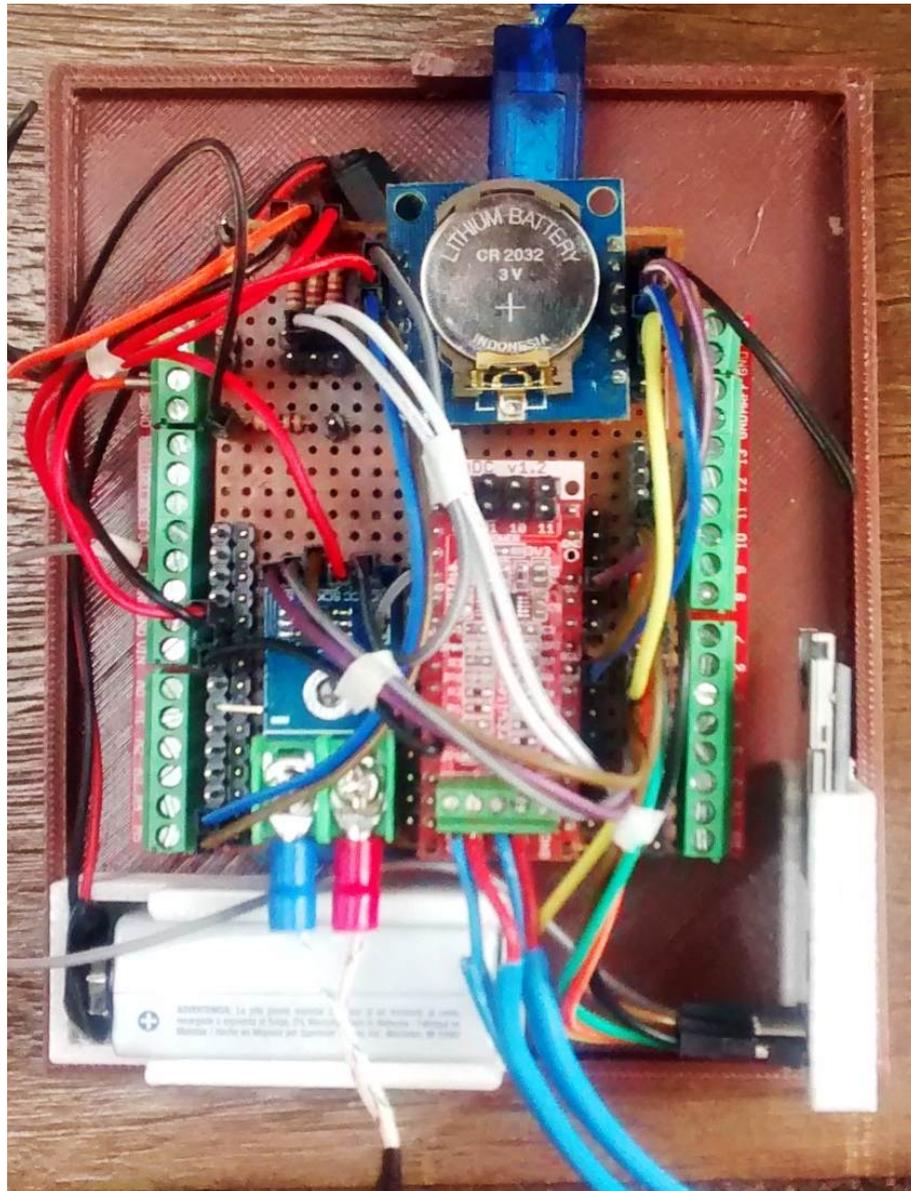


Figura 31: módulo de coleta de dados em funcionamento.

Para a ligação dos sensores e seus complementos à placa Arduino, foi observado a interface de comunicação necessária para cada dispositivo. Desta forma, foi possível integrar sensores diferentes no mesmo pino ou porta, reduzindo o número de portas utilizadas e, assim, possibilitar a conexão de um número maior de sensores.

As interfaces de comunicações usadas são:

- Serial Peripheral Interfase (SPI): protocolo de dados síncronos para comunicação do microcontrolador com os seus periféricos, sendo o microcontrolador rotulado como MASTER e os periféricos como SLAVE. Neste modelo de comunicação usa-se 4 conexões, nomeadas como:
 - MISO (master IN slave OUT) ou SDO (slave data OUT);
 - MOSI (master OUT slave IN) ou SDI (slave data IN);
 - SCK (serial clock);
 - SS (slave select) ou CS (chip select).

- Inter-Integrated Circuit (I2C): trabalha no formato master-slave, onde o master coordena a comunicação com os demais dispositivos. Permite a ligação de até 127 dispositivos slave na mesma conexão, que é feita por dois conectores identificados como SDA (serial data) e SCL (serial clock) sendo que o Arduino vem com os pinos dedicados a essa comunicação. No modelo UNO o pino 4 é usado como SDA e o 5 como SCL. Ao ligar vários dispositivos via I2C deve-se verificar a necessidade do uso de resistor 1,5 k Ω em paralelo com a alimentação positiva e as portas SDA e SCL. Outro fator a ser observado é tamanho máximo dos cabos que, normalmente, é de 1m. Esta distância pode ser maior, desde que se atente com a impedância do sistema. A comunicação I2C identifica os dispositivos através de um endereçamento de 7 bits e o oitavo bit é usado para definir a se a operação é de leitura ou escrita.

- One Wire: é um protocolo de comunicação serial que utiliza somente dois cabos, sendo um com referencia 0V e o outro para alimentação e transmissão de dados. Existe ainda a opção de uso de um cabo adicional dedicado à alimentação. A identificação com os dispositivos slave se dá pelo endereçamento de 64 bits programado de fabrica e impossível de ser alterado. Essa comunicação se comparada a I2C tem menor taxa de transmissão de dados, mas, por outro lado, possui maior alcance.

- Digital por interrupção externa: permite um controle contínuo de uma porta e toda vez que ocorre a alteração do sinal o microcontrolador utiliza a interrupção para priorizar

ou não tarefas e resolver problemas. Neste trabalho o sensor de vazão e anemômetro utilizam este recurso para medir o tempo entre dois pulsos e calcular a frequência. O sensor de direção do vento, por sua vez, o utiliza para identificar alteração na posição e definir o sentido de giro. Para o Arduino UNO as portas 2 e 3 estão definidas para essa função, no entanto é possível definir outras portas para executar essa função.

A tabela 4 é um resumo dos sensores e os locais onde eles estão conectados no módulo, destacando o tipo de comunicação usada em cada sensor. As portas A0, A1, 8 e 9 do módulo de coleta de dados não estão sendo utilizadas. Nestas portas ainda poderia ser conectados outros sensores. Caso ainda sejam necessárias outras portas livres, existe a possibilidade de integrar o leitor de cartão SD (SD Card) ao sensor termopar tipo K, liberando mais duas portas, totalizando 6 portas disponíveis.

Tabela 4: Identificação da ligação dos sensores no Arduino UNO no módulo de coleta de dados.

Portas Arduino UNO	Pinagem Do sensor	Tipo Sensor	Comunicação Usada
A0			
A1			
A2	Sinal 1	Direção do vento	interrupção
A3	Sinal 2		
A4	DAS	Conversor AD e Módulo Calendário	I2C
A5	SCL		
0 – TX			Comunicação Serial com o PC
1 – RX			
2	sinal OUT	Vazão	interrupção
3	sinal OUT	Anemômetro	interrupção
4	SO	Termopar Tipo K	SPI
5	CS		
6	SCK		
7	DQ	Temperatura DS18b20	OneWire
8			
9			
10	SS	SD Card	SPI
11	MOSI		
12	MISO		
13	CSCK		

IV.3.1. Armazenamento dos dados

O armazenamento dos dados é feito no cartão SD, utilizando um arquivo identificado como DADOS.TXT. A saída dos sensores são armazenadas de forma sequencial e separadas por vírgulas. A primeira linha identifica a natureza dos dados ou do sensor que os coletou, formando uma tabela com o cabeçalho de identificação. Este tipo de formato é conhecido como CSV e pode ser facilmente importado em planilhas eletrônicas ou outros programas matemáticos.

Ao ligar o módulo de coleta de dados, ocorre a verificação da presença do cartão SD, para posteriormente criar o arquivo de texto DADOS.TXT. Caso este arquivo já exista no cartão de memória, ele será mantido, mas aberto para inclusão de novos dados a partir da última linha do registro anterior. A figura 32 mostra o formato do armazenamento dos dados no cartão SD, conforme estrutura mostrada na tabela 5. Caso o usuário deseje apagar o arquivo anterior quando ele existe, deve-se incluir o comando:

```
if (SD.exists("dados.txt")) { SD.remove("dados.txt"); }
```

```
data,hora,velocidade_do_vento km/h,vazao em L/h,temp_int,temp_MAX6675,sensor
A0_A1 (V),sensor A2_A3 (V),direcao vento
12.01.2017,15:10:50,1.37,0.00,28.50,31.50,0.000000,0.000000,0,
12.01.2017,15:11:52,1.55,0.00,28.50,31.00,0.000000,0.000000,315,
12.01.2017,15:12:53,2.66,0.00,29.00,31.50,0.000000,0.000000,285,
12.01.2017,15:13:54,0.00,0.00,29.00,31.75,0.000000,0.000000,330,
12.01.2017,15:14:58,1.94,0.00,29.00,31.50,0.000000,0.000000,315,
12.01.2017,15:15:59,1.39,0.00,29.00,32.00,0.000000,0.000000,315,
12.01.2017,15:17:01,0.68,0.00,29.00,32.50,0.000000,0.000000,330,
12.01.2017,15:18:03,2.56,0.00,29.00,32.00,0.000000,0.000000,330,
12.01.2017,15:19:04,2.14,0.00,29.50,32.50,0.000000,0.000000,315,
12.01.2017,15:20:05,0.19,0.00,29.50,32.25,0.000000,0.000000,285,
12.01.2017,15:21:08,2.01,0.00,29.50,32.25,0.000000,0.000000,285,
12.01.2017,15:22:09,0.66,0.00,29.50,32.25,0.000000,0.000000,345,
12.01.2017,15:23:11,1.18,0.00,29.50,32.25,0.000000,0.000000,300,
12.01.2017,15:24:13,1.81,0.00,29.50,32.50,0.000000,0.000000,345,
```

Figura 32: dados coletados e armazenados no cartão SD.

Tabela 5: Dados coletados pelo módulo e armazenado no cartão SD após importados pelo software LibreOfficeCalc.

data	hora	velocidade_do_vento km/h	Vazao L/h	temp_int °C	temp_MAX6675 °C	sensor A0_A1 (V)	sensor A2_A3 (V)	direcao vento em grau
12.01.2017	15:10:50	7,16	0,00	28,50	31,50	0,000000	0,000000	0
12.01.2017	15:11:52	7,72	0,00	28,50	31,00	0,000000	0,000000	315
12.01.2017	15:12:53	11,16	0,00	29,00	31,50	0,000000	0,000000	285
12.01.2017	15:13:54	2,90	0,00	29,00	31,75	0,000000	0,000000	330
12.01.2017	15:14:58	8,93	0,00	29,00	31,50	0,000000	0,000000	315
12.01.2017	15:15:59	7,22	0,00	29,00	32,00	0,000000	0,000000	315
12.01.2017	15:17:01	5,01	0,00	29,00	32,50	0,000000	0,000000	330
12.01.2017	15:18:03	10,85	0,00	29,00	32,00	0,000000	0,000000	330
12.01.2017	15:19:04	9,55	0,00	29,50	32,50	0,000000	0,000000	315
12.01.2017	15:20:05	3,49	0,00	29,50	32,25	0,000000	0,000000	285
12.01.2017	15:21:08	9,14	0,00	29,50	32,25	0,000000	0,000000	285

Após a exportação dos dados para um software de manipulação, como as planilhas eletrônicas, eles podem ser usados para gerar gráficos. A figura 33 mostra um gráfico gerado desta forma. Nesta figura são apresentadas a velocidade e direção do vento, no dia 12 de janeiro de 2017, na cidade de Avaré, nas coordenadas geográficas 23°04'22.5''S 48°55'459'' W, local onde o equipamento foi posicionado.

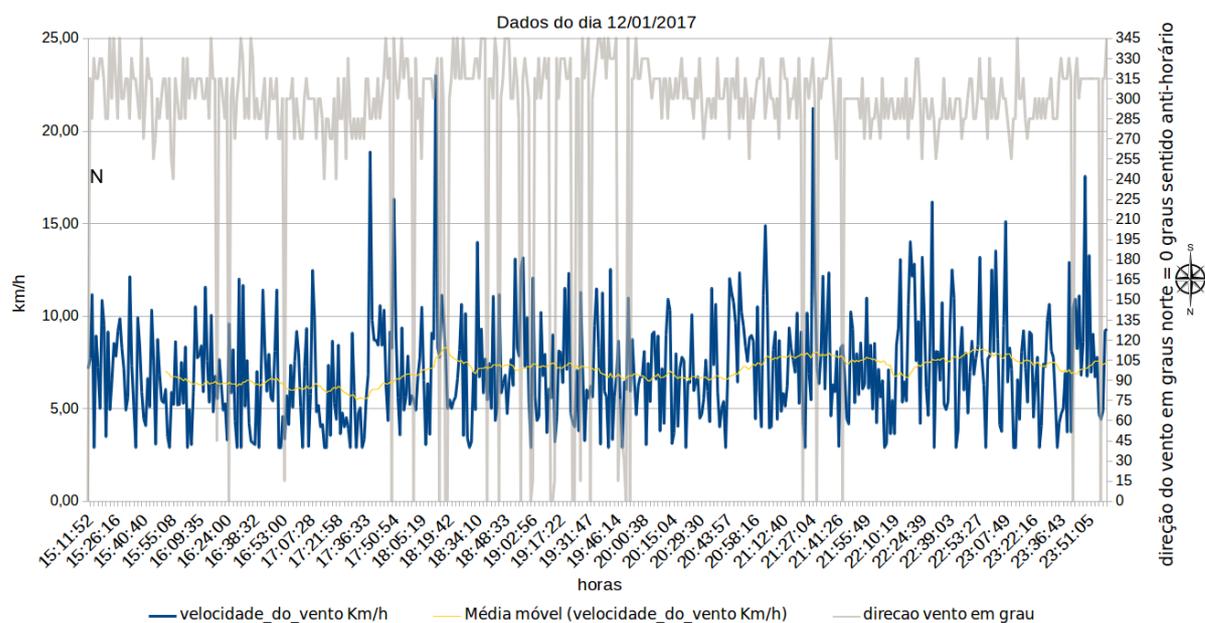


Figura 33: Gráfico com os dados coletados e exportado para LibreOfficeCalc.

V - CONCLUSÕES E TRABALHOS FUTUROS

No presente trabalho foi apresentado o desenvolvimento de um módulo e uma biblioteca para utilização de sensores de medição de velocidade e direção do vento, assim como a integração de outros sensores em um único módulo de coleta e armazenamento de dados. O desenvolvimento do trabalho, permiti concluir que:

- O armazenamento dos dados no cartão SD se mostrou uma forma segura, barata e de fácil de aquisição de dados. A maioria dos computadores, portáteis e de mesa ou até mesmo celulares, possuem dispositivo de leitura desse padrão de cartões.
- A placa Arduino UNO, além de ser de baixo custo se mostrou bastante eficaz na operação do sistema, pois executou a programação de todos os sensores ocupando 86% da sua memória. Deixou espaço ainda para integração de mais alguns sensores que usem outro tipo de comunicação.
- A quantidade de sensores integradas mostrou-se adequada para uma ampla faixa de sensores e, no projeto, ainda ficaram disponíveis as portas A0, A1, 8 e 9. Além da inclusão de demais sensores nestas portas ainda existe a possibilidade de uso de outros com mesmos protocolos de comunicação já usados. Este seria o caso de sensores que utilizam a comunicação I2Cs nas portas A4 e A5 e ONEWIRE na

porta 7. Neste caso, não há necessidade de ocupar as portas livres e os sensores serão integrados aos já instalados.

- O conversor Analógico Digital se mostrou útil para integrar até quatro sensores e apresentou-se também como uma alternativa fácil e precisa de realizar a amplificação do sinal.
- O anemômetro se comportou conforme o esperado e coletou os dados de forma eficaz, se mostrando razoavelmente sensível em baixas velocidades do vento. O ajuste da programação, com um looping de espera do recebimento dos pulsos, mostrou-se fundamental para a determinação das baixas frequência apresentadas nestas condições.
- O uso do módulo com o fornecimento de energia provindo de uma pilha alcalina de 9V se comportou adequadamente, mas apresentou um período de autonomia muito pequeno, de aproximadamente 6 horas. Após este período, a voltagem da bateria ficou abaixo de valores de 6 V e a placa Arduino deixou de operar. Por conta disto, é aconselhável o uso de um conjunto de pilhas com voltagem de 12v ou bateria recarregável de Polímero de Lítio (PoLi) de 11,1 V. Este procedimento possibilita uma autonomia muito maior, caso o sistema não possa ser alimentado diretamente por energia elétrica ou de um computador.

O desenvolvimento do projeto mostrou adequado à proposta original, atingindo o objetivo proposto no estudo e mantendo o foco no baixo custo, porém algumas sugestões para desenvolvimento futuro deste trabalho são:

- Alterar a dimensão do anemômetro de forma a obter uma relação do diâmetro da caneca com o raio do centro de giro da caneca mais adequada. Este procedimento permitiria respostas mais rápidas na aceleração e desaceleração do equipamento, sem sacrificar a leitura em baixas velocidades do vento.
- No sensor de direção do vento, substituir o encoder por um dispositivo sem contato físico, diminuindo o atrito e a resistência ao movimento giratório, possibilitando assim a redução do tamanho do dispositivo. Uma possibilidade é o uso de sensores tipo Reed Switch acionados por magnetismo. Estes dispositivos ainda tem a vantagem do tipo de ligação à placa Arduino, usando uma porta analógica(ao invés de duas portas digitais usadas pelo encorder). Além disto, não

precisa ser monitorado continuamente e exigindo menor demanda da placa. Este fato também diminuiria o consumo de memória e de energia, além eliminar a necessidade de uso de uma posição inicial de referência.

- Fazer uso do módulo de coleta de dados por períodos longos, possibilitando comparar os dados coletados pelo módulo com os dados coletados por instrumentos comerciais, com a finalidade de verificar a repetibilidade e confiabilidade do equipamento em longos períodos de operação.

Toda a programação elaborada para esse módulo, incluindo as bibliotecas são apresentadas no apêndice A e B, do trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

ALCIATORE, DAVID G., HISTAND, M. B., **Introduction to Mechatronics and Measurement Systems**, 4th ed., New York: Mc Graw Hill, 2012.

ANEEL. Resolução normativa nº 482 de 17 de abril de 2012, 17/04/2012. Disponível em: <http://www2.aneel.gov.br/cedoc/ren2012482.pdf>. Acesso em: 10 julho. 2017.

Arduino-Reference. Disponível em: <https://www.arduino.cc/en/Reference/HomePage?from=Reference.Extended>. Acesso em: 20 maio. 2015.

BANJEVIC, M. et al. High-speed CMOS magnetic angle sensor based on miniaturized circular vertical Hall devices. **Sensors and Actuators A: Physical**, v. 178, p. 64–75, maio 2012.

BASEER, M. A. et al. Performance evaluation of cup-anemometers and wind speed characteristics analysis. **Renewable Energy**, v. 86, p. 733–744, fev. 2016a.

BÉGIN-DROLET, A.; RUEL, J.; LEMAY, J. Novel meteorological sensor for anemometer heating control purposes: Part B — Integration into a single sensor. **Cold Regions Science and Technology**, v. 96, p. 53–68, dez. 2013.

CLAROS-MARFIL, L. J.; PADIAL, J. F.; LAURET, B. A new and inexpensive open source data acquisition and controller for solar research: Application to a water-flow glazing. **Renewable Energy**, v. 92, p. 450–461, jul. 2016.

DESIKAN, S. L. N. et al. Fast response co-axial thermocouple for short duration impulse facilities. **Applied Thermal Engineering**, v. 96, p. 48–56, 5 mar. 2016.

DIEESE. **Comportamento das tarifas de energia elétrica no Brasil**. São Paulo, agosto, 2015. (Nota Técnica, 147). Disponível em:

<http://www.dieese.org.br/notatecnica/2015/notaTec147eletricidade.pdf>

DONG-HUI, LI; JING-YU, XU. Investigation of the mixture flow rates of oil-water two-phase flow using the turbine flow meter. **Journal of Physics: Conference Series**, v. 147, number 1, 2009.

DREBUSHCHAK, V. A. Thermocouples, their characteristic temperatures, and simple approximation of the emf vs. T. *Thermochimica Acta*, **Chip Calorimetry**. v. 603, p. 218–226, 10 mar. 2015.

Energia solar pode reduzir consumo elétrico em até 17%. Disponível em: <http://www.mma.gov.br/informma/item/7183-energia-solar-pode-reduzir-consumo-eletrico-em-ate-17>. Acesso em: 6 mar. 2017.

FARRUGIA, R. N.; SANT, T. Modelling wind speeds for cup anemometers mounted on opposite sides of a lattice tower: A case study. **Journal of Wind Engineering and Industrial Aerodynamics**, v. 115, p. 173–183, abr. 2013.

FASINMIRIN, J. T. et al. Development and calibration of a self-recording cup anemometer for wind speed measurement. **African Journal of Environmental Science and Technology**, v. 5, n. 3, p. 212–217, 1 jan. 2011.

GARMABDARI, R. et al. Study on the effectiveness of dual complementary Hall-effect sensors in water flow measurement for reducing magnetic disturbance. **Flow Measurement and Instrumentation**, v. 45, p. 280–287, out. 2015.

HOLMAN, J. P., **Experimental methods for engineers**, 8th ed., New York: McGraw-Hill, 2011.

Inmetro - Tabelas de consumo/eficiência energética. Disponível em: <http://www.inmetro.gov.br/consumidor/pbe/coletores-solares.asp>. Acesso em: 7 mar. 2017.

KOLHARE, N. R, THORAT, P. R.. An Approach Of Flow Measurement In Solar Water Heater Using Turbine Flow Meter. **International Journal of Engineering Research & Technology**. Vol. 2, Issue 1, January- 2013.

LI, D.; XU, J. Measurement of oil-water flow via the correlation of turbine flow meter, gamma ray densitometry and drift-flux model. **Journal Of Hydrodynamics**, v. 27, n. 4, p. 548–555, 2015.

MARIBO PEDERSEN, B. et al. Some experimental investigations on the influence of the mounting arrangements on the accuracy of cup-anemometer measurements. **Journal of Wind Engineering and Industrial Aerodynamics**, v. 39, n. 1, p. 373–383, 1 jan. 1992.

MORAES, C. C.; CASTRUCCI, P.; **Engenharia de Automação Industrial**, 2ª ed. São Paulo: LTC, 2007.

NORTHROB, R. B., **Instrumentation and Measurements**, 2nd ed., Florida: CRC Press, 2005.

N. R. KOLHARE; P. R. THORAT. An approach of flow measurement in solar water heater using turbine flow meter. **International Journal of Computers & Technology**, v. 4, n. 1a, p. 1–4, 2013.

ORLANDO, S.; BALE, A.; JOHNSON, D. A. Experimental study of the effect of tower shadow on anemometer readings. **Journal of Wind Engineering and Industrial Aerodynamics**, v. 99, n. 1, p. 1–6, jan. 2011a.

ORLANDO, S.; BALE, A.; JOHNSON, D. A. Experimental study of the effect of tower shadow on anemometer readings. **Journal of Wind Engineering and Industrial Aerodynamics**, v. 99, n. 1, p. 1–6, jan. 2011b.

PINDADO, S.; CUBAS, J.; SORRIBES-PALMER, F. On the harmonic analysis of cup anemometer rotation speed: A principle to monitor performance and maintenance status of rotating meteorological sensors. **Measurement**, v. 73, p. 401–418, set. 2015.

PINDADO, S.; PÉREZ, J.; AVILA-SANCHEZ, S. On Cup Anemometer Rotor Aerodynamics. **Sensors**, v. 12, n. 5, p. 6198–6217, 10 maio 2012.

RAMOS , F. G., SEIDLER, N., Estudo da energia eólica para aproveitamento em pequenos empreendimentos. **Vivências: Revista Eletrônica de Extensão da URI**, Vol.7, N.13: p.108-127, Outubro 2011.

REHMAN, M. et al. Remote measurement of liquid flow using turbine and fiber optic techniques. **Mechanical Systems and Signal Processing**, v. 25, n. 5, p. 1661–1666, 2011.

RUDTSCH, S.; VON ROHDEN, C. Calibration and self-validation of thermistors for high-precision temperature measurements. **Measurement**, v. 76, p. 1–6, dez. 2015.

SANZ-ANDRÉS et al. Mathematical Analysis of the Effect of Rotor Geometry on Cup Anemometer Response. **The Scientific World Journal**, v. 2014, p. e537813, 3 jul. 2014.

SKEA, A. F.; HALL, A. W. R. Effects of water in oil and oil in water on single-phase flowmeters. **Flow Measurement and Instrumentation**, v. 10, n. 3, p. 151–157, set. 1999.

Sophia Biblioteca - Terminal Web : Disponível em: <http://biblioteca.aneel.gov.br/index.html>. Acesso em: 20 jul. 2017.

THOMAZINI, D., ALBUQUERQUE, P., **Sensores industriais fundamentos e aplicações**, 6ª ed., São Paulo: Erica 2008.

VENKATESHAN, S. P., **Mechanical Measurements**, 2nd ed., West Sussex: John Wiley & Sons Ltd, 2015.

Wind turbine power curves. Disponível em: http://www.wind-power-program.com/turbine_characteristics.htm. Acesso em: 15 jun. 2016.

A - BIBLIOTECAS DO SISTEMA

Sensor de Velocidade do Vento

A biblioteca é composta por dois arquivos, o Anemometro.h, que informa tudo que compõem a biblioteca, e o Anemometro.cpp que é o código fonte onde as funções são implementadas.

Segue abaixo o conteúdo do arquivo Anemometro.h.

```

/*
Anemometro.h - biblioteca que calcula a velocidade do vento em anemometro de 3
caneca
Deve-se informar:
1)A constante de calibração a e b na da equação  $V=a.Fr + b$ 
sendo Fr a frequência de pulsos
2)Numero de pulsos (Npulso) que irá medir e fazer a média da frequência, não
precisa ser necessariamente o numero de pulsos em 1 volta
3)Tempo maximo em ms (tempo) para ler os numeros de pulsos do item 2, excedido esse
tempo a medição é desconsiderado e o programa continua a leitura de outros sensores
4)Qual porta (pino) do arduino o sensor está conectado.
*/
#define Anemometro_h
#include "Arduino.h"

class Anemometro
{
public:
Anemometro(float a, float b,int Npulso, int tempo, int pino);
void calcvelocidade ();

int contador=0;
float totalTime;
float velocidade;
int porta;
private:

```

```

float _a;
float _b;
int _Npulso;
int _tempo;

};
#endif

```

Segue abaixo o conteúdo do arquivo Anemometro.cpp.

```

#include "Arduino.h"
#include "Anemometro.h"

Anemometro::Anemometro(float a, float b, int Npulso, int tempo, int pino)
{
  pinMode(pino, INPUT); // pino do sensor vazão
  _a = a;
  _b = b;
  _Npulso = Npulso;
  _tempo = tempo;
  porta = pino;
}

void Anemometro::calcvelocidade ()
{
  totalTime = 0; //tempo entre os pulsos
  float startTime = 0;
  float tempolimite = 0;

  sei(); //ativa a função interrupção - attachInterrupt(pino,contapulso,
RISING);
  startTime = millis();
  if (contador == 0)
  {
    while (contador == 0) //espera 2 seg para receber algum sinal
      //util para baixas velocidades
      {
        tempolimite = millis()-startTime; //calcula o tempo de inicio da rotina
        if (tempolimite > 2000) //caso não tenha sinal em 2 seg, aborta
          {contador=0; break;}
      }
  }

  if (contador == 1) //inicia a contagem quando receber o primeiro sinal
  {
    startTime = millis();
    while (contador < (_Npulso+1)) //fica em loop até receber N sinais do
sensor hall
    {
      tempolimite = millis()-startTime; //calcula o intervalo entre o
primeiro e //ultimo sinal
      if (tempolimite > _tempo) //caso não tenha N sinais no tempo
determinado, aborda //a contagem
        {contador=0; break;}
    }
  }
}

```

```

detachInterrupt(digitalPinToInterrupt(porta));
if (tempolimito<= _tempo)
{
    totalTime = (millis()-startTime)/(_Npulso); // faz a média do tempo para
intervalo //entre N sinais
    contador = 0; //zera a variavel
}
}
if (contador!=0) {contador =0;}
float frequencia;
if (totalTime>0)
{
    frequencia= (1/totalTime)*1000;
    velocidade = (frequencia * _a + _b);
    if (velocidade<0)
        {velocidade=888888888;}//informa que ocorre algum erro
}
else
{
    velocidade = 0;
}
}
}

```

VELOCIDADE DO VENTO COM EQUAÇÃO GRAU 3

Caso tenha a necessidade usar uma equação de grau 3 ($v=a.x^3+b.x^2+c.x+d$) para calcular a velocidade, usar a biblioteca Anemometro.h e o cabeçalho Anemometro.cpp abaixo.

Segue abaixo o conteúdo do arquivo Anemometro.h para equação de grau 3.

```

/*
Anemometro.h - biblioteca que calcula a velocidade do vento em anemometro de 3
caneca
Deve-se informar:
1)A constante de calibração a,b,c,d na da equação  $V=a.Fr^3+b.Fr^2+c.Fr+d$ 
sendo Fr a frequência de pulsos
2)Numero de pulsos (Npulso) que irá medir e fazer a média da frequência, não
precisa ser necessariamente o numero de pulsos em 1 volta
3)Tempo maximo em ms (tempo) para ler os numeros de pulsos do item 2, excedido esse
tempo a medição é desconsiderado e o programa continua a leitura de outros sensores
4)Qual porta (pino) do arduino o sensor está conectado.
*/
#define Anemometro_h
#include "Arduino.h"

class Anemometro
{
public:
Anemometro(float a, float b,float c, float d,int Npulso, int tempo, int
pino);
void calcvelocidade ();

int contador=0;
float totalTime;
float velocidade;
int porta;
private:
float _a;

```

```

float _b;
float _c;
float _d;
int _Npulso;
int _tempo;

};
#endif

```

Segue abaixo o conteúdo do arquivo Anemometro.cpp para equação de grau 3.

```

#include "Arduino.h"
#include "Anemometro.h"

Anemometro::Anemometro(float a, float b, float c, float d, int Npulso, int
tempo, int pino)
{
  pinMode(pino, INPUT); // pino do sensor vazão
  _a = a;
  _b = b;
  _c = c;
  _d = d;
  _Npulso = Npulso;
  _tempo = tempo;
  porta = pino;
}

void Anemometro::calcvelocidade ()
{
  totalTime = 0; //tempo entre os pulsos
  float startTime = 0;
  float tempolimite = 0;

  sei(); //ativa a função interrupção - attachInterrupt(pino,contapulso,
RISING);
  startTime = millis();
  if (contador == 0)
  {
    while (contador == 0) //espera 2 seg para receber algum sinal
      //util para baixas velocidades
    {
      tempolimite = millis()-startTime; //calcula o tempo de inicio da rotina
      if (tempolimite > 2000) //caso não tenha sinal em 2 seg, aborta
        {contador=0; break;}
    }
  }

  if (contador == 1) //inicia a contagem quando receber o primeiro sinal
  {
    startTime = millis();
    while (contador < (_Npulso+1)) //fica em loop até receber N sinais do
sensor hall
    {
      tempolimite = millis()-startTime; //calcula o intervalo entre o
primeiro e //ultimo sinal
      if (tempolimite > _tempo) //caso não tenha N sinais no tempo
determinado, aborda //a contagem
        {contador=0; break;}
    }
  }
}

```

```

detachInterrupt(digitalPinToInterrupt(porta));
if (tempolimito<= _tempo)
{
    totalTime = (millis()-startTime)/(_Npulso); // faz a média do tempo para
intervalo //entre N sinais
    contador = 0; //zera a variavel
}
}
if (contador!=0) {contador =0;}
float frequencia;
float x3;
float x2;
float x;
if (totalTime>0)
{
    frequencia= (1/totalTime)*1000;
    x = frequencia;
    x2 = x*frequencia;
    x3 = x2*x;
    velocidade = (x3 * _a);
    velocidade = velocidade + x2 * _b;
    velocidade = velocidade + x * _c + _d;
    if (velocidade<0)
        {velocidade=888888888;}
}
else
{
    velocidade = 0;
}
}
}

```

Sensor de direção do Vento

A biblioteca é composta por dois arquivos, o Direcaovento.h, que informa tudo que compõem a biblioteca, e o Direcaovento.cpp que é o código fonte onde as funções são implementadas.

Segue abaixo o conteúdo do arquivo Direcaovento.h

```

#ifndef Direcaovento_h
#define Direcaovento_h
#include "Arduino.h"
#define LATCHSTATE 3

class Direcaovento
{
public:
// ----- Constructor -----
Direcaovento(int pin1, int pin2);

// obtem a posição atual
int getPosition();

// ajusta a nova posição
void setPosition(int newPosition);

```

```

//função chamada pelo tempo ou interrupção para verificar posição //do
encoder.
void tick(void);

private:
int _pin1, _pin2; // pinos do encoder.
int8_t _oldState;
int _position; // Posição interna (4 times _positionExt)
int _positionExt; // Posição externa
};

#endif

```

Segue abaixo o conteúdo do arquivo Direcaovento.cpp

```

#include "Arduino.h"
#include "Direcaovento.h"

//0 array mantem os valores -1 para as entradas decrescentes,
// 1 para as entradas onde a posição aumenta,
// e 0 para as demais posições (sem alteração ou não válidas).

const int8_t KNOBDIR[] = {
0, -1, 1, 0,
1, 0, 0, -1,
-1, 0, 0, 1,
0, 1, -1, 0 };

// ----- Inicialização -----
Direcaovento::Direcaovento(int pin1, int pin2)
{
// fixando pinos do sensor
_pin1 = pin1;
_pin2 = pin2;

// Setup os pinos de entrada
pinMode(pin1, INPUT);
digitalWrite(pin1, HIGH);

pinMode(pin2, INPUT);
digitalWrite(pin2, HIGH);

// enquanto não iniciar o movimento, estado do encoder = 3
_oldState = 3;

// inicia com a posição 0;
_position = 0;
_positionExt = 0;
}

int Direcaovento::getPosition()
{
if (_positionExt == 24)
{
_positionExt = 0;
_position = 0;
}
}

```

```

    if (_positionExt == -24)
    {
        _positionExt = 0;
        _position = 0;
    }
    return _positionExt;
}

void Direcaovento::setPosition(int newPosition)
{
    _position = ((newPosition<<2) | (_position & 0x03));
    _positionExt = newPosition;
}

void Direcaovento::tick(void)
{
    int sig1 = digitalRead(_pin1);
    int sig2 = digitalRead(_pin2);
    int8_t thisState = sig1 | (sig2 << 1);

    if (_oldState != thisState)
    {
        _position += KNOBDIR[thisState | (_oldState<<2)];
        if (thisState == LATCHSTATE)
            _positionExt = _position >> 2;
        _oldState = thisState;
    }
}

```

Sensor de Vazão

A biblioteca é composta por dois arquivos, o Vazao.h, que informa tudo que compõem a biblioteca, e o Vazao.cpp que é o código fonte onde as funções são implementadas.

Segue abaixo o conteúdo do arquivo Vazao.h

```

/*
    Vazao.h - biblioteca que calcula a vazão quando está usando um sensor de
    vazão tipo turbina. Deve-se informar o pino que o sensor está conectado e o valor
    de a e b da calibração => v = a.x+b
    criado em 2016.
    */

#ifndef Vazao_h
#define Vazao_h
#include "Arduino.h"

class Vazao
{
public:
    Vazao(float a, float b,int pulso, int tempo, int pino);
    void calcvazao ();
    int contador;
    int porta;
    float totalTime;
    float vazao;
}

```

```

private:
float _a;
float _b;
int _pulso;
int _tempo;
};

#endif

```

Segue abaixo o conteúdo do arquivo Vazao.cpp

```

#include "Arduino.h"
#include "Vazao.h"

Vazao::Vazao(float a, float b, int pulso, int tempo, int pino)
{
  pinMode(pino, INPUT); // pino do sensor vazão
  _a = a;
  _b = b;
  _pulso = pulso;
  _tempo = tempo;
  porta = pino;
}

void Vazao::calcvazao ()
{
  totalTime = 0; //tempo entre os pulsos
  float startTime = 0;
  float tempolimite = 0;
  sei();
  if (contador == 1) //inicia a contagem quando receber o
  { //primeiro sinal
    startTime = millis();
    while (contador < _pulso) //fica em loop até receber o numero
    // de pulsos informado
    {
      tempolimite = millis()-startTime; //calcula o tempo a
      //partir do 1º pulso
      if (tempolimite > _tempo) //caso não tenha os pulsos
      //informado no tempo informado, abandona a contagem (caso a
      //vazão pare)
      {contador=0; break;}
    }
    detachInterrupt(digitalPinToInterrupt(porta));
    if (tempolimite <= _tempo)
    {
      totalTime = (millis()-startTime)/(_pulso-1);
      // faz a média do tempo para intervalo entre N sinais
      contador = 0; //zera a variavel
    }
  }

  if (contador>1) {contador =0;}
  float frequencia;
  if (totalTime>0)
  {
    frequencia= (1/totalTime)*1000;
    vazao = (frequencia * _a + _b);
  }
}

```

```
    }  
    else  
    {  
        vazao = 0;  
    }  
}s
```

B - EXEMPLO DE PROGRAMAÇÃO PARA USO DO MÓDULO DE COLETA DE DADOS.

Para uso do módulo de coleta de dados, deve-se carregar as bibliotecas do sensor a ser usado e também as bibliotecas de comunicação do módulo com a sensor, observando que existe diversas formas de comunicação e deve-se verificar qual a usado por cada sensor.

Segue abaixo o modelo de programação a ser exportada para a placa Arduino UNO, permitindo o uso dos sensores citados neste trabalho.

```
//bibliotecas de comunicação
#include <Ethernet.h>
#include <Dhcp.h>
#include <EthernetServer.h>
#include <EthernetUdp.h>
#include <Dns.h>
#include <EthernetClient.h>
#include <OneWire.h>//comunicação OneWire
#include <SPI.h>//para módulo SDcard
#include <Wire.h>

//bibliotecas dos sensores
#include <Direcaovento.h> // inclui a biblioteca do sensor direcao do vento
#include <DS1307.h>//biblioteca do modulo calendario e hora
#include <Anemometro.h> // inclui a biblioteca do anemometro
#include <Vazao.h> // inclui a biblioteca vazão
#include <DallasTemperature.h>//sensor de temperatura dallas DS18B20
#include <SD.h>// módulo SDcard
#include <max6675.h>
#include <Nanoshield_ADC.h>
#define ONE_WIRE_BUS 7// sensor DS18B20 conectado porta 7
```

```

//portas e variaveis do encoder direção vento *****
Direcaovento encoder(A2, A3);
static int Pos = 0; //posição inicial
int newPos ;

// Inicia o relógio*****
DS1307 rtc(SDA, SCL);

//dados do anemometro*****
Anemometro anemometro(1.1009,1.2234,3,5000,3);
int porta; //anemometro porta=3,sensor vazão porta=2

//dados da vazao*****
Vazao vazao(7.27,3.645,5,5000,2);

//arquivo SDCard*****
File myFile;

//define as portas do MAX6675*****
int thermoS0 = 4;
int thermoCS = 5;
int thermoSCK = 6;
MAX6675 thermocouple(thermoSCK, thermoCS, thermoS0);

//classe do conversor AD*****
Nanoshield_ADC adc;

//sensor temperatura dallas ds18b20*****
OneWire oneWire(ONE_WIRE_BUS);//comunicação Onewire com qualquer
//dispositivo ONEWIRE, incluindo ds18b20
DallasTemperature sensors(&oneWire);//passa referencia OneWire para
// sensor dallas

void setup()
{
  // inicia a comunicação com o computador
  Serial.begin(9600);

  //*****direção do vento*****
  //Habilita a função interrupção nos pinos analógicos(Port C)
  PCICR |= (1 << PCIE1);
  /* Habilita a interrupção do pino 2 (01)e pino 3 (11) das
  portas analógicas habilitada acima (port C)*/
  PCMSK1 |= (1 << PCINT10) | (1 << PCINT11);

  //***** inicia o relógio*****
  rtc.begin();
  // ativo o relógio no modo de execução
  rtc.halt(false);
  // inicia biblioteca do Dallas DS18B20 acoplado a ele
  sensors.begin();
  //abre ou cria arquivo TXT

  *****SD card*****
  while (!Serial)

```

```

{ ; } /* espera conexão serial */
Serial.print(F("abrindo cartao SD..."));

if (!SD.begin(10)) /* verifica a abertura do cartão conectado ao pino 10 */
{
Serial.println(F("erro na abertura do cartao!"));
return;
}
//desmarque a linha abaixo caso queira apagar arquivo anterior
/*if (SD.exists("dados.txt")) { SD.remove("dados.txt"); } */
Serial.println(F("cartao pronto.));
myFile = SD.open("dados.txt", FILE_WRITE);

if (myFile)
{
Serial.println(F("escrevendo em dados.txt...));
Serial.println(F("data,hora,velocidade_do_vento(m/s),vazao_em_L/h,temp_int,
t emp_MAX6675,sensor A0_A1 (V),sensor A2_A3 (V),direcao vento"));
myFile.println("data,hora,velocidade_do_vento(m/s),vazao_em_L/h,temp_int,te
mp_MAX6675,sensor A0_A1 (V),sensor A2_A3 (V),direcao vento");
myFile.close();
Serial.println(F("gravado.));
}
else
{
// se o arquivo nao abrir, avise:
Serial.println(F("erro ao abrir dados.txt));
Serial.println(F("dados somente na tela"));
Serial.println(F("data,hora,velocidade_do_vento(m/s),vazao_em_L/h,temp_int,t
emp_MAX6675,sensor A0_A1 (V),sensor A2_A3 (V),direcao vento"));
}
}

// ativa interrupção do pino A0 ao A5-semelhante attachInterrupt
// encoder.tick()tem a função de :
// armazenar e verificar o sinal dos pinos A2 e A3
// caso a posição tenha sido alterada, atualiza a posição
ISR(PCINT1_vect) { encoder.tick(); }

```

void loop()

```

/* a rotina abaixo lê a posição do vento e transmite para o usuario*/
{
static int pos = 0; //posição inicial
int newPos = encoder.getPosition();
//transforma a informação da posição em graus - sentido horário
if (newPos > 0){ newPos = newPos * 15;}
//transforma a informação da posição em graus,sentido anti-horário
if (newPos < 0){ newPos = (newPos * 15)+360;}

//*****envia a data e hora*****
myFile = SD.open("dados.txt", FILE_WRITE);
if (myFile)
{
myFile.print(rtc.getDateStr());
myFile.print(",");
myFile.print(rtc.getTimeStr());
myFile.print(",");
}
}

```

```

Serial.print(F("data:"));
Serial.println(rtc.getDateStr());
Serial.print(F("hora:"));
Serial.println(rtc.getTimeStr());
}
else
{
Serial.print(F("data:"));
Serial.println(rtc.getDateStr());
Serial.print(F("hora:"));
Serial.println(rtc.getTimeStr());
}
//*****anemometro*****
porta=3;
attachInterrupt(digitalPinToInterrupt(anemometro.porta),contapulso, RISING);
anemometro.calcvelocidade();

if (myFile)
{
myFile.print (anemometro.velocidade,4);
myFile.print(",");
Serial.print (F("velocidade media do vento: "));
Serial.println (anemometro.velocidade,4);
Serial.println (anemometro.totalTime,4);
//Serial.println (anemometro.contador,4);//caso queira ver os pulsos
contados
//Serial.println (anemometro.totalTime,4);//caso queira ver o tempo medio
dos
//Serial.println (anemometro.contador,4);//caso queira ver os pulsos
contados
//Serial.println (anemometro.frequencia,4);//caso queira ver a frequencia
}
else //caso não tenha o cartao SD mostrará no computador
{
Serial.print (F("velocidade media do vento: "));
Serial.println (anemometro.velocidade);
//Serial.println (anemometro.contador,4);//caso queira ver os pulsos
contados
//Serial.println (anemometro.totalTime,4);//caso queira ver o tempo medio
dos
//Serial.println (anemometro.contador,4);//caso queira ver os pulsos
contados
//Serial.println (anemometro.frequencia,4);//caso queira ver a frequencia
}
//delay(1000); /*desmarque essa linha se desejar pause após leitura do
anemometro*/

//*****vazao*****
porta=2;
attachInterrupt(digitalPinToInterrupt(vazao.porta),contapulso, RISING);
vazao.calcvazao();

if (myFile)
{
myFile.print (vazao.vazao);
myFile.print(",");
Serial.print (F("vazao media: "));
Serial.println (vazao.vazao);
}
else
{

```

```

Serial.print (F("vazao media: "));
Serial.println (vazao.vazao);
}

//*****sensor temp. Dallas DS18B20*****
sensors.requestTemperatures();//comando para obter temperatura
if (myFile)
{
myFile.print(sensors.getTempCByIndex(0));
myFile.print(",");
Serial.print (F("temperatura interna: "));
Serial.print (sensors.getTempCByIndex(0));
Serial.println (" C");
}
else
{
Serial.print (F("temperatura interna: "));
Serial.print (sensors.getTempCByIndex(0));
Serial.println (" C");
}

//*****temp. MAX6675*****
delay(300);//tempo para estabilidade do sensor
if (myFile)
{
myFile.print(thermocouple.readCelsius());
myFile.print(",");
Serial.print (F("temperatura MAX6675: "));
Serial.print (thermocouple.readCelsius());
Serial.println (" C");
}
else
{
Serial.print (F("temperatura MAX6675 "));
Serial.print (thermocouple.readCelsius());
Serial.println (" C");
}

//*****sensores conectados ao conversor AD*****
if (myFile)
{
//voltage entre porta A0 e A1-para uso de radiometro
myFile.print(adc.readDifferentialVoltage01(),6);
myFile.print(",");
//voltage entre porta A2 e A3-para uso de radiometro
myFile.print(adc.readDifferentialVoltage23(),6);
myFile.print(",");

Serial.print (F("Voltage A0/A1: "));
Serial.print(adc.readDifferentialVoltage01(),6);
Serial.println (" v");
Serial.print (F("Voltage A2/A3: "));
Serial.print(adc.readDifferentialVoltage23(),6);
Serial.println (F(" v"));
}
else
{
Serial.print (F("Voltage A0/A1: "));
Serial.print(adc.readDifferentialVoltage01(),6);
Serial.println (F(" v"));
}

```

```

Serial.print (F("Voltagem A2/A3: "));
Serial.print(adc.readDifferentialVoltage23(),6);
Serial.println (F(" v"));
}

//*****direcao do vento*****

if (myFile)
{
myFile.print(newPos);
myFile.println(",");
myFile.close();
Serial.print (F("direcao vento: "));
Serial.println (newPos);
Serial.println (F("*****"));
}
else
{
Serial.print (F("direcao vento: "));
Serial.println (newPos);
Serial.println (F("*****"));
}
if (pos != newPos)
{
pos = newPos;
}

//*****tempo para nova leitura dos sensores*****
//delay(60000); //ajuste o tempo – nesse exemplo 1 minuto
}

void contapulso () //função que conta os pulsos do sensor hall
{
if (porta ==2) {vazao.contador++;} //se estiver usando sensor de vazão
//retire a barra dupla antes do if.
if (porta ==3) {anemometro.contador++;} //se estiver usando anemometro
//retire a barra dupla antes do if.
}

```

C - REPRESENTAÇÃO ELÉTRICA DO MÓDULO DE COLETA DE DADOS

Durante a confecção do módulo de coleta de dados foi priorizado a conexão dos componentes visando o uso da quantidade mínima de portas do microcontrolador e a disposição dos componentes sobre a placa tem o objetivo de agrupá-los, ocupando menor espaço e reduzir o tamanho do equipamento.

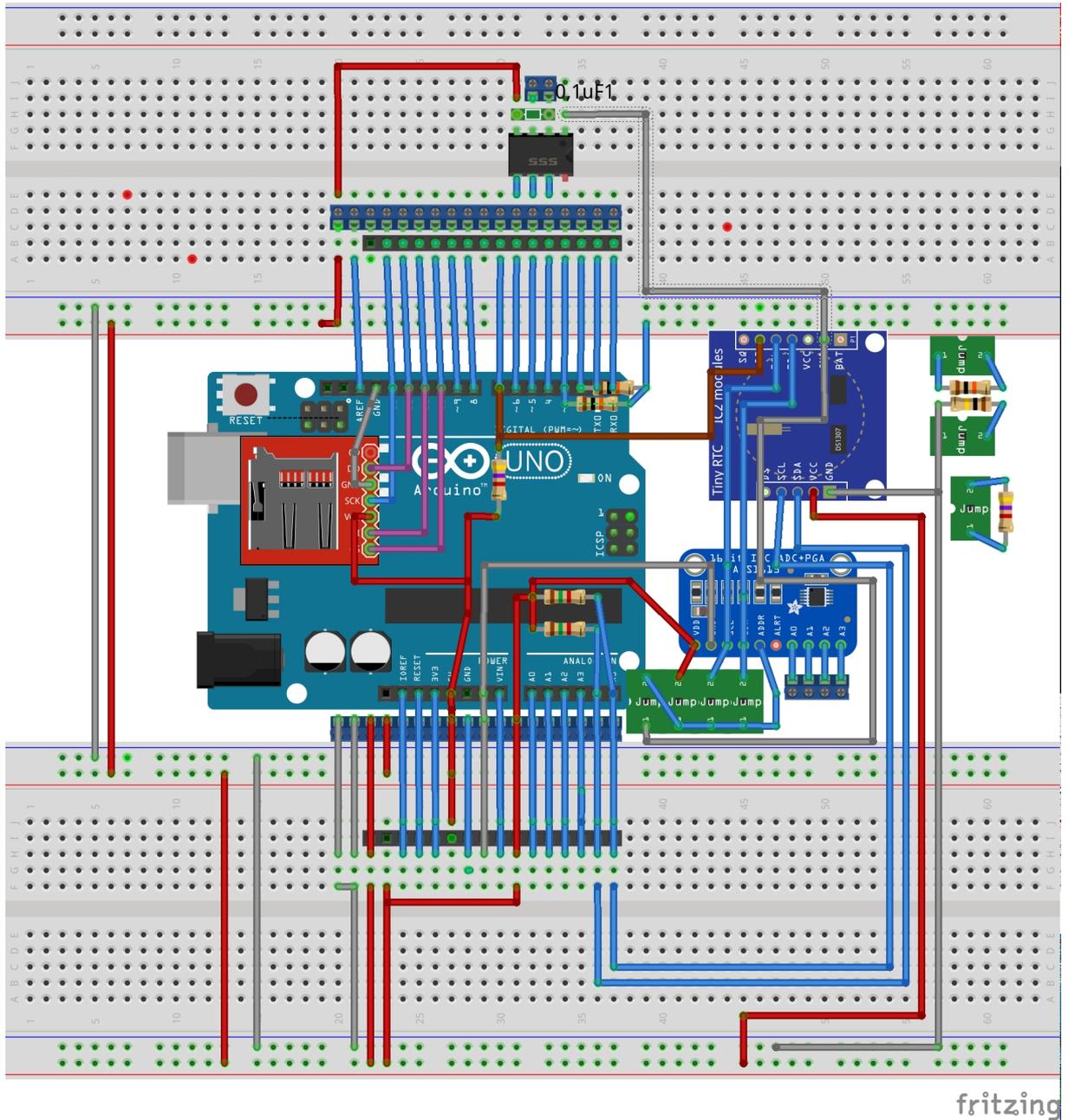


Figura 34: Ligação elétrica dos componentes do módulo representado na placa protoboard.

Segue abaixo as figuras que representam a ligação dos componentes, a confecção da placa de circuito impresso, em tamanho real, das trilhas de cobre e da impressão por silkscreen da identificação dos componentes.

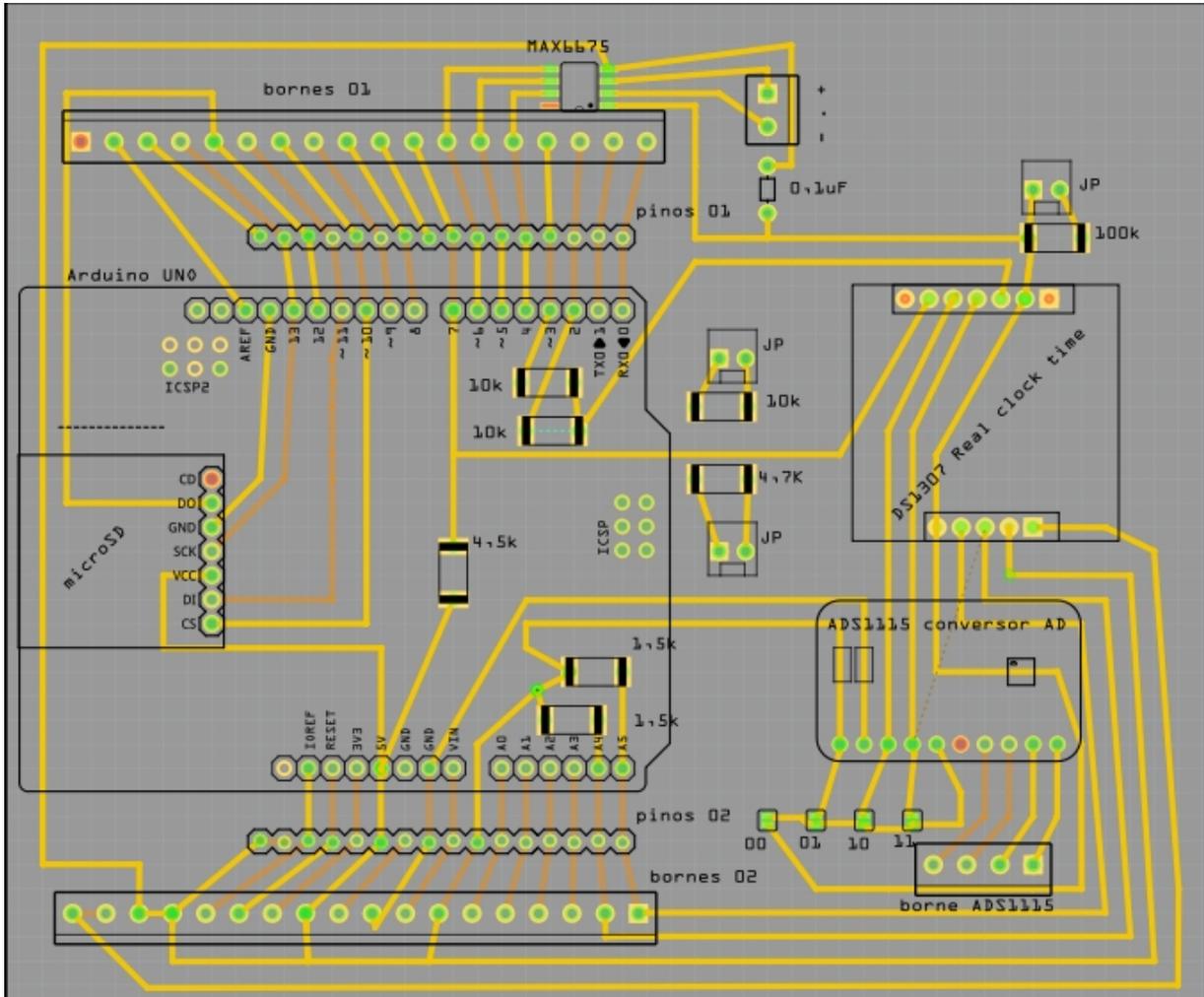


Figura 35: Placa de circuito impresso do módulo de coleta de dados: trilha de cobre e identificação dos componentes.

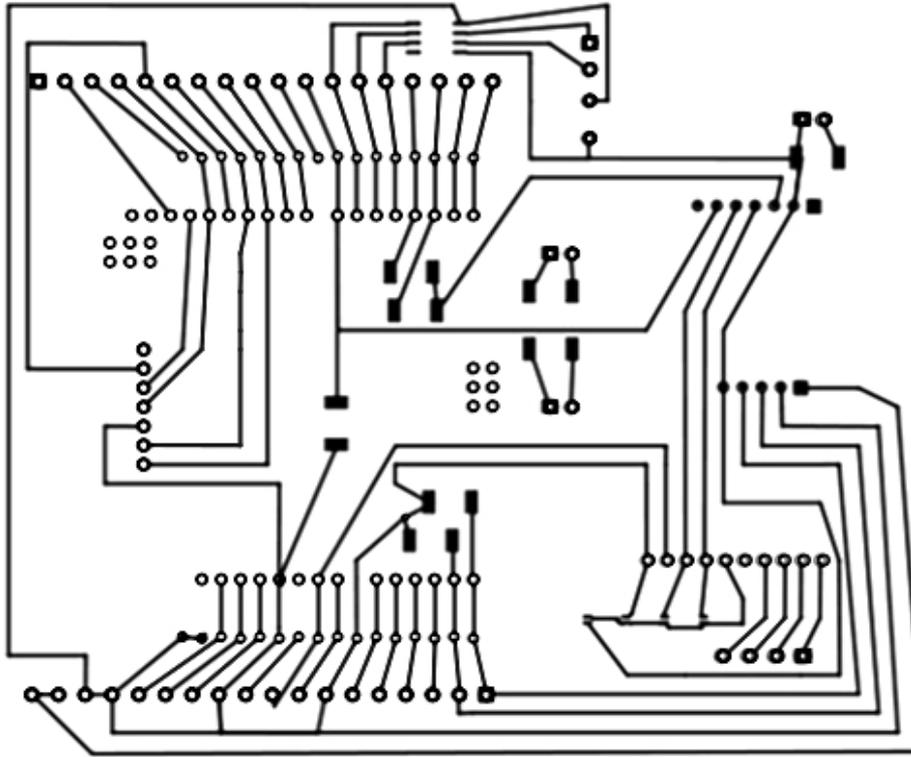


Figura 36: Placa de circuito impresso do módulo de coleta de dados: trilhas de conexão dos componentes.

