**Universidade do Estado do Rio de Janeiro**
Centro de Ciencias e Tecnologia
Faculdade de Engenharia

Gustavo Pessanha Alvim
Pedro da Costa Di Marco

# Projeto e construção de uma estação meteorológica aplicada a uma embarcação teleoperada

Rio de Janeiro

2016

Gustavo Pessanha Alvim
Pedro da Costa Di Marco

# Projeto e construção de uma estação meteorológica aplicada a uma embarcação teleoperada

Projeto de graduação apresentado, como requisito parcial para obtenção de grau de Engenheiro Eletricista, à Faculdade de Engenharia da Universidade do Estado do Rio de Janeiro.

Orientador: Prof. Dr. José Paulo Vilela Soares da Cunha

Rio de Janeiro

2016

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

A475 Alvim, Gustavo Pessanha.
Projeto e construção de uma estação meteorológica aplicada a uma embarcação teleoperada / Gustavo Pessanha Alvim, Pedro da Costa Di Marco. – 2015.
129f.

Orientador: José Paulo Vilela Soares da Cunha.
Projeto Final (Graduação) - Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.
Bibliografia p. 101-103

1. Engenharia elétrica. 2. Estação meteorológica. 3. *Software* de monitoramento. I. Cunha, José Paulo Vilela Soares da. II. Marco, Pedro da Costa Di. III. Universidade do Estado do Rio de Janeiro. IV. Título.

CDU 621.3

Gustavo Pessanha Alvim

Pedro da Costa Di Marco

**Projeto e construção de uma estação meteorológica aplicada a uma embarcação teleoperada**

Projeto de graduação apresentado, como requisito parcial para obtenção de grau de Engenheiro Eletricista, à Faculdade de Engenharia da Universidade do Estado do Rio de Janeiro.

Aprovada em 18 de janeiro de 2016

Banca Examinadora:

---

Prof. Dr. José Paulo Vilela Soares da Cunha

Faculdade de Engenharia – UERJ

---

Prof. Dr. Douglas Mota Dias

Faculdade de Engenharia – UERJ

---

Prof. Dr. Orlando Bernardo Filho

Faculdade de Engenharia - UERJ

Rio de Janeiro

2016

## AGRADECIMENTOS

Agradeço a Deus, inteligência suprema, causa primária de todas as coisas.

Aos meus pais Helvio Pinheiro Alvim e Neiva Pessanha Alvim, meus irmãos Felipe Pessanha Alvim e Juliana Pessanha Alvim e à minha família como um todo, por estarem sempre ao meu lado.

Ao orientador José Paulo Vilela Soares da Cunha pelo auxílio, dedicação e paciência que foram determinantes para conclusão do projeto.

Aos meus chefes, da época em que estive trabalhando pela Technip, Paulo Vaz, Carlos Pinheiro da Silva e Cristina Araújo pela compreensão, flexibilidade e incentivo nos diversos momentos em que, talvez, poderiam não ser tão compreensivos, mas foram mesmo assim.

Aos amigos de trabalho da empresa Technip, pela amizade e incentivo.

Aos amigos feitos na UERJ, desde o primeiro período até o último. Foram essenciais para o encerramente deste ciclo.

Aos amigos feitos no curso técnico do CEFET/RJ que estiveram comigo até hoje.

Ao meu amigo e dupla de projeto Pedro da Costa Di Marco. Tive o prazer de começar esta amizade no curso técnico e desde então seguimos juntos nessa caminhada.

À todos que de alguma forma contribuíram para a realização deste projeto.

GUSTAVO PESSANHA ALVIM

## AGRADECIMENTOS

Primeira a Deus, por toda força e renovação de fé nos momentos em que estes foram abaladas ao longo do curso.

À minha família Antionio Di Marco, Benedita da Costa Di Marco, Felipe da Costa Di Marco, Debora Chiara Di Marco e ao pequeno anjo Sophia Chiara Di Marco pelo apoio constante e confian inabalável de que conseguiria conquistar mais este objetivo.

Ao orientador José Paulo Vilela Soares da Cunha por primeiramente dar-me a oportunidade de ser seu bolsista e durante o projeto, o seu apoio.

Aos amigos Celso Pereira, Rodrigo Belford, Daniela Coelho, Simone Cordeiro, Adriane Duarte, Pedro Victor de Oliveira e Paulo Victor que tive a oportunidade de conhecer e que nos apioamos para realizar este objetivo comum.

Aos amigos resistentes do primiro período Gabriel Viana, Thiago Carvalho e Vinicius Louzada e a todos os outros que participaram também nesta caminhada.

Ao Sergio Felix por todo o apoio em momentos cruciais e pela inclusão no Billops Engenharia e por saber que ganhei um irmão para o resto da vida.

Ao William Cunha, Matheus Vargas e Alexandre Braucks por mantermos a amizade de ensino médio e saber que será para o resto da vida.

À Hellen Rigo, por tudo sempre, onde não existem palavras para descrever. A Mariana Vieitos também.

À minha dupla e irmão Gustavo Pessanha, por me dar esta oportunidade de concluirmos o que começamos no curso técnico e por estarmos juntos desde então e saber que este laço durará para sempre.

PEDRO DA COSTA DI MARCO

# RESUMO

ALVIM, Gustavo Pessanha; DI MARCO, Pedro da Costa. *Projeto e Construção de uma Estação Meteorológica aplicada a uma Embarcação Teleoperada*. Projeto (Graduação em Engenharia Elétrica com ênfase em Sistemas Eletrônicos) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2016.

Neste projeto foi desenvolvida uma estação meteorológica que será aplicada a uma embarcação não tripulada, com fins de monitoração ambiental. Esta estação é composta de sensores que medem quatro variáveis, que são temperatura, pressão, umidade e índice pluviométrico. É descrita a escolha de cada componente, que foi realizada por meio de um estudo comparativo que avaliaou as características das alternativas. Neste projeto também foi desenvolvido um *software* de monitoramento destas mesmas variáveis através de comunicação sem fio. Este programa tratará da exibição atual das medidas, medidas mínimas e máximas durante seu funcionamento, geração de gráficos e armazenamento dos dados. As instruções para a montagem e funcionamento de todo o sistema são descritas ao longo do texto.

**Palavras-chave:** Estação Meteorológica, Instrumentação, Sensores, *Software* de Monitoramento.

## ABSTRACT

ALVIM, Gustavo Pessanha; DI MARCO, Pedro da Costa. *Projeto e Construção de uma Estação Meteorológica aplicada a uma Embarcação Teleoperada*. Projeto (Graduação em Engenharia Elétrica com ênfase em Sistemas Eletrônicos) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2016.

In this project a weather station that will be applied to a unmanned surface vessel, with environmental monitoring purposes was developed. This station consists of sensors that measure four variables, which are temperature, pressure, humidity and rainfall. The selection of each component is described, which was carried out by a comparative study and the evaluation of the characteristics of alternatives. This project developed monitoring software of these variables through wireless communication. This software shows the current values of the measurements, minimum and maximum measurements, graphics and data storage. The instructions for assembly and operation of the entire system are described throughout the text.



**Keywords:** Weather Station, Instrumentation, Sensors, Monitoring Software.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## ABREVIAÇÕES

| | |
|---|---|
| ADC | *Analog-to-digital converter* |
| CSV | *Comma-Separated Values* |
| EEPROM | *Electrically-Erasable Programmable Read-Only Memory* |
| GFSK | *Gaussian frequency-shift keying* |
| GPIO | *General Purpose Input/Output* |
| I2C | *Inter-Integrated Circuit* |
| IDE | *Integrated Development Environment* - Ambientes de Desenvolvimento Integrado |
| ISM | *Industrial Scientific and Medical* |
| ITU | *International Telecommunication Union* |
| I/O | *Input/Output* |
| HDMI | *High-Definition Multimidia Interface* |
| PC | *Personal Computer* – Computador Pessoal |
| PWM | *Pulse-Width Modulation* – Modulação por largura de Pulso |
| RAM | *Random Access Memory* – Memória de acesso aleatório |
| SD | *SanDisk* |
| SPI | *Serial Peripheral Interface* |
| SQL | *Structured Query Language* |
| UART | *Universal Asynchronous Receiver/Transmitter* |
| USB | *Universal Serial Bus* |
| WLAN | *Wireless Local Area Network* |
| Wi-Fi | *Wireless Fidelity* |

**SUMÁRIO**

# 1. INTRODUÇÃO

Todos se interessam pelo clima. Em diversos países, boletins de previsão do tempo são os mais populares programas de televisão. Pessoas em todo o mundo precisam saber como será o tempo hoje ou amanhã para que possam saber as possibilidades de semeadura, plantio e colheita, viagens marítimas ou por outro meio de transporte, fazer preparativos contra perigos naturais eminentes, como furacões. Querem saber as condições para a prática de esportes ao ar livre ou atividades recreativas ou, simplesmente, o que vestir ou se é necessário levar consigo um guarda-chuva.

Os Serviços Meteorológicos Nacionais observam o tempo e o clima de forma contínua, fornecendo um fluxo regular de dados que são transmitidos ao redor do mundo com o propósito de previsões e planejamento. O tempo não respeita fronteiras nacionais, e o trabalho realizado por meteorologistas, comumente nos bastidores, para nosso benefício e segurança é um trabalho de equipe.

As observações e dados ajudam a criar produtos e previsões. Desta maneira, podemos fornecer serviços confiáveis e efetivos para apoiar a segurança de propriedades e vidas, bem como o bem-estar geral da população. Exemplos são operações seguras, regulares e eficientes de aviação, agricultura, pesca e segurança alimentar, segurança marítima, monitoramento de recursos hídricos e avisos antecipados de desastres naturais e preparação da comunidade [1].

A aquisição das informações do tempo é realizada por estações meteorológicas, objeto deste projeto.

## 1.1. Objetivo

O objetivo deste projeto é desenvolver, construir e testar uma estação de medições meteorológicas que será aplicada a uma embarcação tele-operada para aquisição de dados remotamente. Essa estação medirá, armazenará e transmitirá os dados de temperatura, pressão, umidade relativa do ar e índice pluviométrico.

## 1.2. Organização do texto

Este texto foi organizado da seguinte maneira: O capítulo 2 foi reservado para breve introdução teórica sobre estações meteorológicas e apresentação das variáveis meteorológicas medidas pela estação.

O capítulo 3 foi destinadado ao estudo e definições dos componentes para utilização no projeto. Primeiramente é feita a apresentação dos diversos componentes estudados para aplicação no projeto e em seguida apresentamos a escolha justificada do componente.

O capítulo 4 apresenta o projeto do protótipo, desde os sensores e módulos utilizados até o sistema de vedação da estação com a caixa estanque e seu acoplamento ao barco.

O capítulo 5 apresenta o *software* de aquisição dos dados, assim como sua interface com o microcontrolador Arduino. Neste capítulo constam também os códigos e linguagens utilizadas assim como o programa de monitoração meteorológica.

O capítulo 6 é relativo aos testes e calibração dos sensores e módulo de comunicação.

O capítulo 7 apresenta os testes com o protótipo montado e operando de forma integrada, analisando os dados colhidos pela estação e realizando uma discussão sobre os resultados.

Finalmente no capítulo 8 apresentamos as conclusões deste trabalho.

## 2. ESTAÇÕES METEOROLÓGICAS

A adaptação para a sobrevivência dos seres vivos do planeta sempre esteve ligada aos recursos naturais existentes e disponíveis. O sucesso do crescimento de determinada espécie também está relacionado à sua adaptação e à disponibilidade desses recursos. Devido a estas afirmações surgiu o interesse em conhecer e levantar a disponibilidade de recursos e por isso sempre procurou desenvolver formas de medir e registrar as condições da natureza. Datas de floração das cerejeiras e data de congelamento dos lagos foram registradas visando conhecer as condições reinantes já nos séculos VII e X. Observações dos astros e medidas das estações do ano já eram conhecidas por alguns povos há mais de 3000 anos.

Desta busca surgiram as primeiras intenções de parametrizar as condições de tempo, visando conhecer condições em que a vida se desenvolve melhor, muitas vezes tentando explorar da melhor forma os recursos naturais.

No ano de 170 d.C., o grego Claudius Galenus of Pergamum, foi preconizador de uma escala de temperatura que dava um valor para água fervente e um valor para o gelo, porém o primeiro termômetro foi idealizado por Galileo Galilei, por volta do início do século XVII. O equipamento era feito com um tubo de vidro com um bulbo preenchido com vinho. Com o ar retirado deste tubo, antes de colocar o vinho, o líquido se expandia, subindo ou descendo, conforme a temperatura deste líquido. Com a colocação de escala, pelo colega de Galileu, Sanctorius Sanctorius, estava criado o primeiro termômetro semelhante aos existentes atualmente.

Com a aprimoração do equipamento por vários pesquisadores, somente no início do século XVIII se fez uma leitura compreensível, usando vinho como líquido e o congelamento da água como ponto fixo. Foram propostas várias escalas para as leituras, mas somente em 1742 Anders Celsius propôs de se graduar o termômetro com 100 graus para o ponto de ebulição da água e zero grau para o ponto de derretimento da neve.

Outro equipamento de parametrizar tempo e clima também foi criado no século XVII,desenvolvido por Torricelli, discípulo de Galileu, para tentar mostrar a existência de vácuo na atmosfera, o cientista verificou que uma coluna de mercúrio dentro de um tubo de vidro permanecia 76 cm acima do reservatório do metal. O mesmo verificou que a altura da coluna variava ao longo do dia, do ano e conforme as condições de tempo. Estas relações foram observadas e outro sensor meteorológico, para medir pressão atmosférica, foi desenvolvido.

Equipamentos dos mais diversos foram construídos, porém muitas das medições, ou observações necessitavam de uma pessoa para as leituras, pois os registradores surgiram algum tempo depois. Assim surgiram os observadores meteorológicos que tinham como tarefa realizar e registrar os diferentes parâmetros em horários pré-determinados, normalmente às 9, 15 e 21 horas.

Além das medidas de temperatura do ar, solo, chuva, umidade relativa do ar, insolação, velocidade e direção do vento, ocorrência de granizo, formação de geada, nebulosidade, visibilidade, evaporação, ocorrência de orvalho, entre outras, foram formadas grandes bases de dados, a partir de tais informações, que auxiliaram aos técnicos no estudo e na definição dos vários tipos de clima existentes.

Ainda hoje a existência do observador meteorológico é importante no auxílio aos meteorologistas para previsão de tempo, que é uma informação cada vez mais importante para a logística da vida moderna. Porém, com o aparecimento e desenvolvimento de equipamentos de funcionamento eletrônico, como o sensor termoelétrico termopar, caracterizado como um sensor de temperatura, desenvolvido em 1821. Assim, a estação meteorológica automática foi e está tomando o lugar deste profissional. As dificuldades inerentes à profissão juntamente com as facilidades dos equipamentos eletrônicos, fizeram com que o observador cada vez mais se esteja afastando da estação, causando uma transformação da informação gerada aos pesquisadores. Equipamentos que medem e registram a maioria das informações mais importantes do clima, e podendo registrá-los em pequenos espaços de tempo permitem a obtenção de informações antes nunca imagináveis de serem geradas. A estação automática praticamente não apresenta limite de armazenamento de dados e a existência de softwares facilita a manipulação e exploração destes dados [2].

Uma estação meteorológica consiste na utilização de sensores para análise de parâmetros meteorológicos. As Estações Meteorológicas são utilizadas em todo o mundo como ferramenta de registro a curto, médio e longo prazo dos fenômenos meteorológicos para que sejam feitos os estudos e análises necessários, como a classificação do clima de um local ou região, estudos sobre incidências de ventos, etc.

De acordo com o tipo de equipamentos que abrigam, as estações meteorológicas podem ser classificadas em convencionais e automáticas. Nas primeiras, a coleta dos dados é feita manualmente por técnicos. As segundas são dotadas de sensores eletrônicos e de meios de transmissão dos dados para uma central. As estações podem ser classificadas também de acordo com a aplicação na qual é utilizada. Cada estação possui os sensores de acordo com sua aplicação e podem medir variáveis como pressão atmosférica, temperatura, umidade relativa do ar, precipitação, radiação solar, direção e velocidade do vento, etc.

Os diversos sensores existentes nas estações são nada mais que dispositivos eletrônicos que realizam as medições. Os sensores são dispositivos sensíveis a algum tipo de energia do ambiente (luminosa, térmica, cinética) e que relaciona informações sobre uma grandeza que precisa ser medida (temperatura, pressão, velocidade, corrente, etc).

O uso das estações meteorológicas automáticas, que há algum tempo atrás somente era viável para grandes universidades e empresas, hoje é realidade no meio rural, onde produtores já se beneficiam de suas informações em tempo real e sem limites de distância entre a coleta e a leitura das medidas. Hoje o produtor rural pode acompanhar as condições de tempo de sua propriedade independente do local onde ele se encontre. As estações, por estas características, possibilitam a aplicação e o desenvolvimento de novas tecnologias, como o momento exato de aplicação de agroquímicos, dependendo das condições instantâneas do tempo, com operações mais limpas e tecnicamente mais corretas em várias áreas da produção agropecuária [3].

Os principais órgãos operacionais de meteorologia do Brasil que mantêm uma rede de observação em nível nacional são: O Instituto Nacional de Meteorologia (INMET), do Ministério da Agricultura, Pecuária e Abastecimento; o Departamento de Controle do Espaço Aéreo (DECEA) do Comando da Aeronáutica e a Diretoria de Hidrografia e Navegação (DHN) do Comando da Marinha, ambos do Ministério da Defesa, além do Instituto Nacional de Pesquisas Espaciais do Ministério da Ciência e Tecnologia (INPE).

## 2.1. Variáveis Medidas

As variáveis escolhidas para medição da estação meteorológica foram:

- Temperatura - Medição da temperatura em graus Celsius (°C).
- Umidade do Ar - Medição da umidade relativa do ar - de modo indireto - em porcentagem (%).
- Pressão Atmosférica - Medição da pressão atmosférica em hectopascal (hPa).
- Milímetros lineares - Medição da quantidade de precipitação, neste caso, chuva (mm).

As variáveis foram escolhidas por serem as mais importantes de uma estação meteorológica e muitas análises e previsões podem ser feito pela comparação destas medidas ao longo de determinados períodos de tempo. A indicação do vento não será feita porque sua implementação apresentou alto custo.

## 2.2. Aplicação das Estações Meteorológicas

Na agricultura ou na indústria, esses equipamentos são aplicados para monitorar as condições meteorológicas em curtos intervalos de tempo, com objetivos imediatos. Entre eles, o de verificar a ocorrência de ventos fortes ou temperaturas médias e máximas diárias ou ainda variações na intensidade e direção dos ventos.

Outra importância das estações meteorológicas é a previsão de possíveis tragédias naturais, evitando desastres ainda maiores e salvando vidas se detectadas a tempo.

Já para as estações meteorológicas localizadas no mar, como é o caso do projeto em questão, as informações obtidas serão de alta importância para portos, empresas *offshore* (alto-mar), pescadores, pesquisadores em projetos científicos, etc [4].

A figura 1 apresenta exemplos destas estações.

(a)

(b)

Figura 1 – (a) Típica estação meteorológica terrestre; (b) Estação Meteorológica embarcada a uma bóia.
Extraído de [1] e [4].

# 3. ESTUDO E SELEÇÃO DE COMPONENTES

Este capítulo é destinado ao estudo dos componentes disponíveis no mercado e seleção do melhor componente encontrado para aplicação no projeto.

Apresentamos inicialmente os componentes separados para análise e uma descrição técnica dos mesmos. Em seguida apresentamos de uma forma mais completa o componente que foi escolhido para utilização no projeto, assim como os motivos de sua escolha.

O primeiro componente estudado podemos dizer que é o cérebro do projeto, apresentamos os microcontroladores, responsáveis por receber as informações dos sensores e processar as informações.Em seguida são estudados todos os sensores meteorológicos da estação e finalizando este capítulo são apresentados os tipos de comunicação entre a estação e o computador utilizado para análise das variáveis.

## 3.1. Estudo dos Microcontroladores

Os principais microcontroladores estudados para utilização no projeto foram o Arduino UNO e o Raspberrry PI *model* B.

### *3.1.1. Arduino UNO*

O Arduino é uma plataforma de prototipação com um microcontrolador Atmel, com funcionamento da placa de 5 V, proveniente de um cabo USB (*Universal Serial Bus*) ou de com alimentação externa de 6 a 20V, sendo o ideal para o seu funcionamento de 7 a 12V. Como o Arduino raramente trabalha sozinho, existindo um terminal que fornece alguns pinos de alimentação, terra, referência de tensão, reset para os *shields* e/ou componentes que podem vir a ser usados em conjunto com o Arduino. Possui alguns pinos digitais que podem ser usados como entrada e saída, e entre esses pinos digitais, existem os que podem ser usados como saídas PWM (*Pulse-Width Modulation*), comunicação serias e interrupção externa. Existem também entradas analógicas, onde cada uma tem resolução de 10 bits. Possui um esquema de programação, feito através de comunicação serial, muito simplificado, no qual consegue-se iniciar sua programação de maneira bem simplificada. O necessário para começo do projeto com este microcontrolador é um computador com o *software* instalado e um cabo USB para a transmissão de dados entre as plataformas operacionais.

*3.1.2. Raspberrry PI model B*

O Raspberry se propõe a ser um computador completo, de baixo custo. Para seu controle e funcionalidade precisamos conectar a ele um monitor, um teclado e um mouse.

Começar a criação do projeto no Raspberry seria um pouco mais demorado, já que é necessário baixar a imagem do sistema operacional, copiar essa imagem para o cartão SD (SanDisk), instalar o Raspbian, que é uma distribuição baseado em Linux, configurar o sistema operacional, e conseqüentemente começar a usar a interface de entradas e saídas de uso geral (GPIO - *General Purpose Input/Output*) disponíveis para os seus projetos. No conector de GPIO, existem 26 pinos, e tem-se acesso a pinos I2C *(Inter-Integrated Circuit)*, SPI (*serial peripheral interface*), UART (*Universal Asynchronous Receiver/Transmitter*) e um pino PWM. A placa contém saídas de áudio e vídeo, além de entradas ethernet para RJ45, HDMI (*High-Definition Multimidia Interface*) e para cartão SD. Inicialmente, seria um trabalho mais demorado e trabalhoso para começo do projeto com este microcontrolador.

## 3.2. Escolha do Microcontrolador

O Microcontrolador Arduino UNO foi o escolhido para utilização no projeto, a escolha por este componente se deu após análise dos diversos microcontroladores existentes.

A plataforma Arduino UNO, exibido na figura 2, consegue atender perfeitamente a interface com os sensores escolhidos para o projeto, assim como a aquisição e transmissão dos dados fornecidos [5]. Dentre os principais motivos da escolha deste microcontrolador seguem os principais os fatores:

- Possui *hardware* mínimo apropriado para o projeto;
- Alta capacidade de processamento de sinais;
- Quantidade de pinos de dados adequada;
- Facilidade de conexão com os outros componentes;
- Melhor relação custo benefício;
- Disponibilidade do mercado;


Figura 2 – Arduino UNO.
Extraído de [6].

A plataforma Arduino possui um microcontrolador que consome pouca energia e que dá ao usuário completo controle de seu hardware. Através de uma IDE (*Integrated Development Environment*) própria, é possível escrever programas que fazem interface com vários tipos de dispositivos como sensores, atuadores, visores, outros microcontroladores, além de possibilitarem uma forma rápida de desenvolvimento de protótipos. Sua característica de baixo consumo de energia foi o diferencial para a sua escolha.

O componente principal da placa Arduino UNO é o microcontrolador ATMEL ATMEGA328, um dispositivo de 8 bits da família AVR com arquitetura RISC avançada e com encapsulamento DIP28. Ele conta com 32 kB de memória *flash*, onde 512 *bytes* são utilizados pro *bootloader*, 2 kB de RAM (*Random Access Memory*) e 1 kB de EEPROM (*Electrically-Erasable Programmable Read-Only Memory*). Pode operar a até 20 MHz, porém na placa Arduino UNO opera em 16 MHz, valor do cristal externo que está conectado aos pinos 9 e 10 do microcontrolador.

Possui 28 pinos, sendo que 23 desses podem ser utilizados como I/O (*Input/Output*). A Figura 3 exibe a pinagem do microcontrolador ATMEGA328.

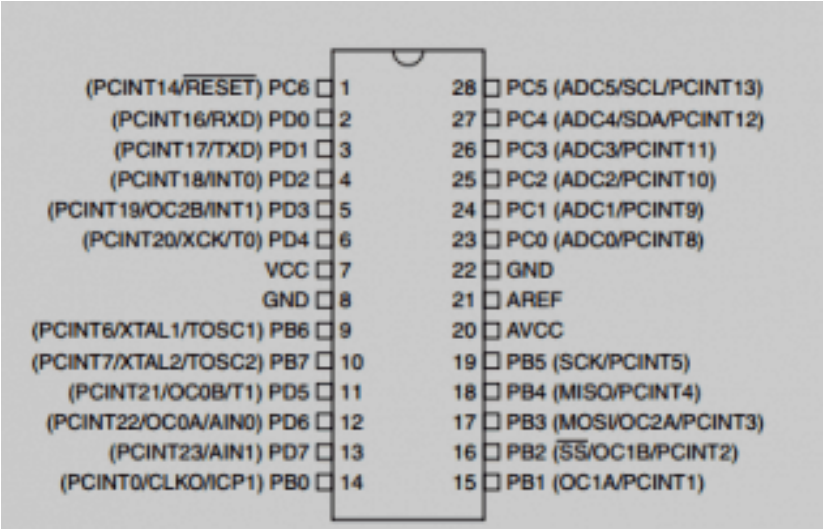



Figura 3 – Pinagem ATMEGA328 usado no Arduino UNO.
Extraído de [6].

A placa pode ser alimentada pela conexão USB ou por uma fonte de alimentação externa. A alimentação externa é feita através do conector *Jack* com polo positivo no centro, onde o valor de tensão da fonte externa deve estar entre os limites 6 a 20 volts, porém se alimentada com uma tensão abaixo de 7 V, a tensão de funcionamento da placa, que no Arduino Uno é 5 V, pode ficar instável e quando alimentada com tensão acima de 12 V, o regulador de tensão da placa pode superaquecer e danificar a placa. Dessa forma, é recomendado para tensões de fonte externa valores de 7 a 12 Volts.

Quando o cabo USB é plugado a um PC (*Personal Computer*), por exemplo, a tensão não precisa ser estabilizada pelo regulador de tensão. Dessa forma a placa é alimentada diretamente através do cabo USB. O circuito apresenta alguns componentes que protegem a porta USB do computador em caso de alguma anormalidade.

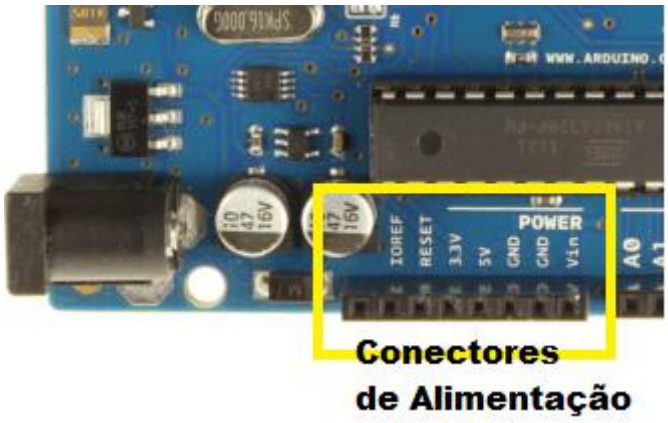



Figura 4 – Conectores de alimentação Arduino UNO.
Exraído de [5].

Os conectores disponíveis no microcontrolador para alimentação são exibididos na Figura 4 e uma breve descrição é mostrada abaixo:

- **IOREF -** Fornece uma tensão de referência para que *shields* possam selecionar o tipo de interface apropriada, dessa forma *shields* que funcionam com a placas Arduino que são alimentadas com 3,3V podem se adaptar para ser utilizados em 5V e vice-versa.
- **RESET -** pino conectado a pino de RESET do microcontrolador. Pode ser utilizado para um reset externo da placa Arduino.
- **3,3 V.** - Fornece tensão de 3,3V para alimentação de *shield* e módulos externos. Corrente máxima de 50 mA.
- **5 V -** Fornece tensão de 5 V para alimentação de *shields* e circuitos externos.
- **GND -** pino de referência, terra.
- **VIN -** pino para alimentar a placa através de *shield* ou bateria externa. Quando a placa é alimentada através do conector *Jack*, a tensão da fonte estará nesse pino.

A placa Arduino UNO possui pinos de entrada e saídas digitais, assim como pinos de entradas e saídas analógicas. Na Figura 5 é exibida a pinagem conhecida como o padrão Arduino.

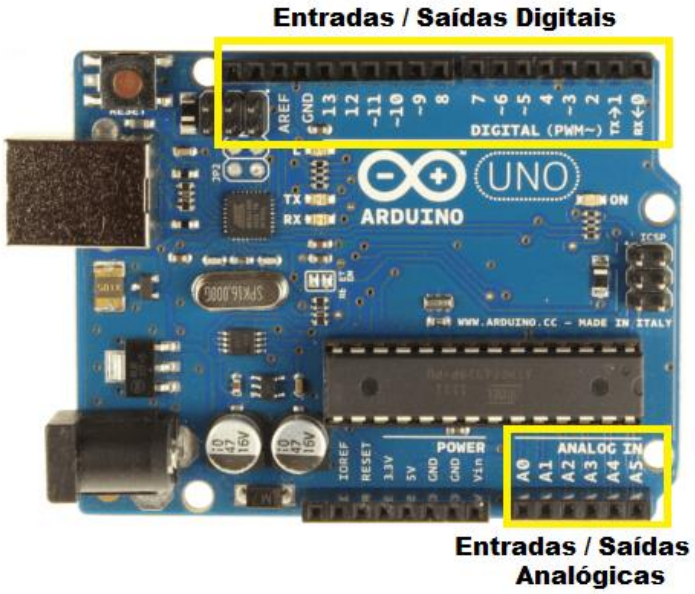



Figura 5 – Pinos de entrada e saída no Arduino UNO.
Extraído de [5].

Conforme exibido na Figura 5, a placa Arduino UNO possui 14 pinos que podem ser usados como entrada ou saída digitais (de 0 a 13. Estes pinos operam em 5 V, onde cada pino pode fornecer ou receber uma corrente máxima de 40 mA. Cada pino possui resistor de *pull-up* interno que pode ser habilitado por *software*. Alguns desses pinos possuem funções especiais:

- **PWM** - Pinos 3, 5, 6, 9, 10 e 11 podem ser usados como saídas PWM de 8 bits.
- **Comunicação serial** - Pinos 0 e 1 podem ser utilizados para comunicação serial. Deve-se observar que estes pinos são ligados ao microcontrolador responsável pela comunicação USB com o PC;
- **Interrupção externa -** Pinos 2 e 3 podem ser configurados para gerar uma interrupção externa.

Para interface com o mundo analógico, a placa Arduino UNO possui 6 entradas, onde cada uma tem a resolução de 10 bits. Por padrão a referência do conversor AD (*analog-to-digital converter*) está ligada internamente a 5 V, ou seja, quando a entrada estiver com +5 V o valor da conversão analógica digital será 1023. O valor da referência pode ser mudado através do pino AREF.

Quem manipula a placa e projeta o circuito que será conectado aos seus I/O's deve ter muito cuidado, pois, entre os pinos do microcontrolador e a barra de pinos, não há nenhum

resistor, que limite a corrente, além disso, dependendo do local onde está trabalhando pode-se provocar curto circuito nos pinos já que a placa não possui isolação na sua parte inferior.

A placa não conta com botão liga/desliga, ou seja, se desejarmos desligar a alimentação, deve-se retirá-la.

Resumo das principais características do Arduino UNO:

- Tamanho: 5,3 cm x 6,8 cm x 1,0 cm
- Microcontrolador: ATmega328
- Tensão de operação: 5 V
- Tensão de entrada (recomendada): 7 V a 12 V
- Tensão de entrada (limites): 6 V a 20 V
- Pinos de entradas/saídas (I/O) digitais: 14 (dos quais 6 podem ser saídas PWM)
- Pinos de entradas analógicas: 6
- Corrente DC por pino I/O: 40 mA
- Corrente DC para pino de 3,3V: 50 mA
- Memória Flash: 32 kB
- Velocidade de Clock: 16 MHz
- Temperatura de operação: de 10º a 60 ºC

## 3.3. Estudo dos Sensores Meteorológicos

### *3.3.1. Medidor de Umidade do Ar - Higrômetro*

Um higrômetro é um instrumento que serve para medir a umidade presente nos gases, mais especificamente na atmosfera. É utilizado principalmente em estudos do clima, mas também em locais fechados onde a presença de umidade excessiva ou abaixo do normal poderia causar danos, por exemplo, em peças de museus, documentos de bibliotecas e elementos de laboratórios.

O físico suíço Horace Benedict de Saussure inventou o primeiro higrômetro no final dos anos 1700. Ele descobriu que certos materiais orgânicos se expandem quando expostos a umidade e se contraem quando expostos a ausência de umidade. Horace prendeu um dos lados de um fio de cabelo em uma parte fixa e o outro lado foi preso a um braço de alavanca, que é tracionado suavemente por uma mola. Com o aumento da umidade, o cabelo esticava e movia o braço.

Atualmente a maioria dos higrômetros mecânicos, exemplificado na Figura 6, utilizam uma mola levemente enrolada que possui um revestimento absorvente de umidade em um de seus lados.Assim que o material absorve umidade ele se expande e faz com que a mola gire.


Figura 6 - Higrômetro mecânico.
Extraído de [35]

Nos dias de hoje, existem diversos tipos de sensores de umidade. Os sensores de umidade mais comuns são os capacitivos e muitos deles utilizam a mesma tecnologia: Um material dielétrico especial é prensado entre duas placas, formando um capacitor. O material dielétrico absorve umidade e muda sua capacitância em função da umidade absorvida. Os sensores capacitivos são de baixo custo e razoavelmente fiéis. O maior problema é que a capacitância não pode ser medida diretamente, é preciso convertê-la para uma frequência ou tensão. Os sensores de umidade resistivos medem as alterações na resistência ou impedância de um material higroscópico. Muitos dos sensores resistivos de umidade relativa utilizam um polímero condutor ou uma superfície coberta por um sal com dois eletrodutos. Com o aumento da umidade, a superfície tratada absorve a umidade e a resistência diminui [7].

Alguns dos exemplos dos sensores de umidade do ar pesquisados são exibidos na Figura 7.


(a) (b) (c)

Figura 7 – Sensores de umidade do ar: (a) DHT22; (b) HIH-4030; (c) SHT15. Extraído de [36].

Algumas comparações entre os sensores de umidade são feitas na Tabela 1.

*Tabela 1 – Comparação de características dos sensores de umidade do ar.*

| | DHT22 | HIH-4030 | SHT15 |
|---|---|---|---|
| **Faixa de Op.** | 0 a 100% | 0 a 100% | 0 a 100% |
| **Precisão** | ± 2% | ± 3,5% | ± 2% |
| **Resolução** | 0,1% | 0,1% | 0,05% |

*3.3.2. Medidor de Temperatura - Termômetro*

Os sensores de temperatura são dispositivos que provocam variações elétricas no material sensor em função de variações na temperatura. Isto provoca variações na tensão, resistência ou corrente de saída do dispositivo. Estas variações podem ser aferidas e mostradas, ou gravadas em um computador.

Diversas escalas podem ser utilizadas para expressas temperaturas. Entre elas estão a Fahrenheit, a Celsius e a Kelvin. Na maior parte dos países, a escala Celsius é a mais empregada. A escala Fahrenheit é bastante empregada nos países de língua inglesa, em especial nos Estados Unidos da América e Inglaterra. A escala Kelvin é mais utilizada para fins científicos.

Os termômetros mais antigos, como o que é apresentado na figura 8, utilizam expansão térmica como base para a medição da temperatura. Como exemplo pode-se citar os termômetros de bulbo, que são empregados para medir temperatura corporal, temperatura interna e externa em residências e temperaturas de líquidos. Este tipo de termômetro consiste em um tubo de vidro graduado, com um reservatório de líquido, geralmente mercúrio ou álcool. Logo que a temperatura

aumenta/diminui, o líquido começa a se expandir/comprimir e é forçado a subir/descer pelo tubo, marcando assim a temperatura.

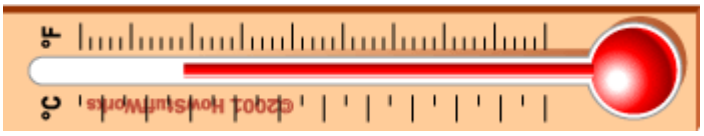


Figura 8 - Termômetro de bulbo.
Extraído de [37].

Pares termoelétricos, exibidos na Figura 9, são alguns dos mais antigos sensores de temperatura. Eles funcionam através do princípio que Thomas Seebeck descobriu por volta dos anos 1800. Se dois metais diferentes são unidos, eles provovam uma pequena tensão. Esta tensão é proporcional a temperatura: quanto mais quente a junção ficar mais alta será a tensão produzida.

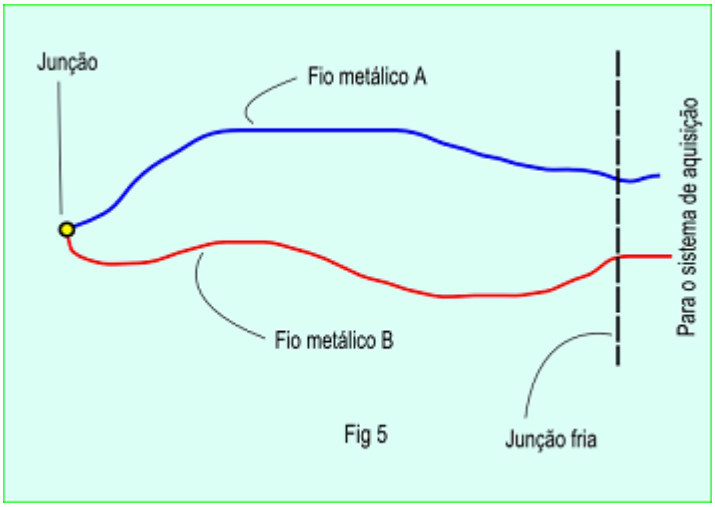


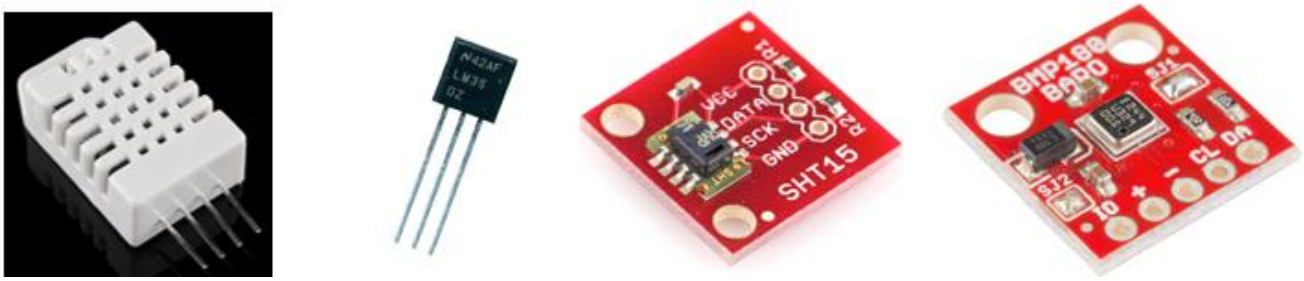

Figura 9 - Esquemático da junção de um termopar.
Extraído de [38].

Tão antigos quanto os pares termoelétricos, os termistores foram descobertos por Michael Faraday por volta dos anos 1800. A palavra termistor vem da contração das palavras “térmico” e “resistor”. Termistores são, portanto, dispositivos em que a resistência varia com a temperatura e são empregados em estações meteorológicas, termômetros digitais e em qualquer situação em que se deseja monitorar a temperatura. Existem dois tipos de termistores: os de coeficiente negativo de temperatura e os de coeficiente positivo de temperatura. Nos termistores de coeficientes negativos,

a resistência diminui com o aumento da temperatura, já nos termistores de coeficientes positivos, a resistência aumenta com o aumento da temperatura [8].

Figura 10 - Termistores
Extraído de [36].

Alguns dos exemplos dos sensores de temperatura pesquisados estão mostrados na Figura 11.




(a) (b) (c) (d)

Figura 11 – Sensores de temperatura: (a) DHT22; (b) LM35; (c) SHT15; (d) BMP180.
Extraído de [36].

Algumas comparações entre os sensores são feitas na Tabela 2.

*Tabela 2 – Comparação de características dos sensores de temperatura.*

| | **DHT22** | **LM35** | **SHT15** | **BMP180** |
|---|---|---|---|---|
| **Faixa de Op.** | -40 a 80 °C | -55 a 150 °C | -40 a 124 °C | -40 a 85 °C |
| **Precisão** | ± 0,5 °C | ± 0,5 °C | ± 0,5 °C | ± 0,5 °C |
| **Resolução** | 0,1 °C | 0,1 °C | 0,04 °C | 0,1 °C |

*3.3.3. Medidor de Pressão Atmosférica – Barômetro*

O barômetro é o instrumento na meteorologia usado para medir a pressão atmosférica. A mudança de pressão pode prever mudanças de curto prazo no clima. A contínua medição da pressão atmosférica é usada em sistemas de analise temporal de superfície.

A pressão atmosférica é a pressão que o ar da atmosfera exerce sobre a superfície do planeta. As unidades de pressão representam o comprimento de uma coluna de mercúrio suficiente para equilibrar a pressão atmosférica, sendo as mais utilizadas o milímetro de mercúrio (mmHg) e a polegada de mercúrio (inHg). As unidades de pressão vêm do experimento de Torricelli, realizado em 1643. Virando uma proveta cheio de mercúrio, com sua abertura para baixo, em uma cuba também com Mercúrio, sem permitir a entrada de ar, a coluna irá se estabilizar em uma determinada altura h, onde neste momento, o sistema entrou em equilíbrio.

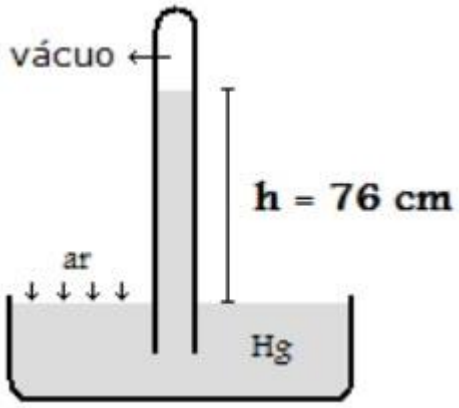



Figura 12 - Demostrativo da experiência de Torricelli.
Extraído de [39].

Torricelli demonstrou que a altura h vale 760 mm, de modo que o valor 760 mmHg (29,2 inHg) é adotado como uma atmosfera padrão.

Barômetros mecânicos, conforme mostrado na Figura 13, medem a pressão atmosférica absoluta pela comparação desta com o vácuo. Uma pequena peça de metal que é capaz de medir esta pressão é projetada para expandir-se e contrair-se no vácuo. Esta peça é chama de fole. Um dos lados do fole é mantido fixo, o outro lado é ligado a um braço de alavanca, com o propósito de amplificar o pequeno movimento de fole. O braço da alavanca é posteriormente conectado a um ponteiro em um mostrador. Assim que a pressão atmosférica cai, o fole se expande, causando um deslocamento no ponteiro [9].


Figura 13 - Barômetro mecânico.
Extraído de [40].

Os sensores de pressão barométricos trabalham de maneira similar aos barômetros mecânicos. Ao invés de fole, um pequeno diafragma é montado em cima de uma câmara de vácuo. O diafragma é fabricado com um material piezo-resistivo, que muda sua resistência em função da variação de pressão.

Alguns dos exemplos dos sensores de pressão pesquisados são apresentados na Figura 14:


(a) (b)

Figura 14 – Sensores de pressão atmosférica: (a) BMP180; (b) MS5803.
Extraído de [36].

Algumas comparações entre os sensores de pressão atmosférica são feitas na Tabela 3.

*Tabela 3 – Comparação de características dos sensores de pressão atmosférica.*

| | **BMP180** | **MS5803** |
|---|---|---|
| **Faixa de Op.** | 300 a 1100 hPA | 0 a 14 bar |
| **Precisão** | ± 0,12 hPa | ± 0,2 bar |
| **Resolução** | 0,01 hPa | 0,001 bar |

A conversão de unidades é de 0,001 hectopascal = 1 bar.

### *3.3.4. Medidor de Precipitação - Pluviômetro*

O pluviômetro é o medidor usado na meteorologia para recolher e indicar, em milímetros lineares, a quantidade de líquidos ou sólidos precipitados, ou seja, chuva, neve ou granizo. Medido em milímetros, é o somatório da precipitação num determinado local durante um intervalo de tempo estabelecido.

O projeto mais comum encontrado na construção dopluviômetro é do tipo gangorra, onde a chuva atravessa um funil, enchendo um recipiente, e quando atinge o limite, este balanço vira, despeja a água numa área conhecida e começa a acumular água no segundo recipiente [10]. Esta quantidade de água despejada é um valor em milimetros conhecido, e quando despejado, passa por um contador, onde este acumula a quantidade de vezes que a gangorra despejou água. O índice pluviométrico é dado pela quantidade armazenada no contador vezes a área conhecida inicialmente.

Existe um projeto semelhante onde o pluviômetro, que utiliza o mesmo princípio de gangorra, é baseado em um sistema de um imã magnético que passa em um interruptor, fazendo circular corrente e armazenado em um contador. O princípio de funcionamento é muito semelhante com o descrito acima, onde o índice pluviométrico também é dado pelo valor do contador vezes a área da gangorra exibido na Figura 15 [10].


Figura 15 – Pluviômetro do tipo gangorra.
Extraído de [36]

O projeto escolhido, de acordo com as opções estudadas, foi com a utilização de um sensor ultrassônico, que mede a distância entre o alvo e o sensor. Foi escolhido este tipo por simplicidades da montagem mecânica Neste projeto, colocaríamos sensor no topo do recipiente, provavelmente cilíndrico por ser mais comumente encontrado, e colocaríamos algum refletor no fundo do recipiente, sendo feito de um material flutuante na presença de água, como o isopor ou madeira. Então este sensor mediria a distância até o refletor, subtrairia da altura total entre o sensor e o fundo do recipiente quando vazio, encontrando assim a quantidade de água que está no recipiente, que é a altura que o refletor se elevou com a água. Esta altura é o valor do índice pluviométrico em milímetros lineares.

Outro projeto semelhante a este descrito acima, é a utilização de um sensor de infravermelho. O processo de aquisição e tratamento de dados seria análogo ao sensor de distância.

Todos estes projetos utilizariam um recipiente com área de perímetro conhecida e necessitariam de uma eletroválvula para liberação da água do recipiente em um intervalo de tempo pré-determinado ou de acordo com um comando remoto, apresentado na Figura 16.

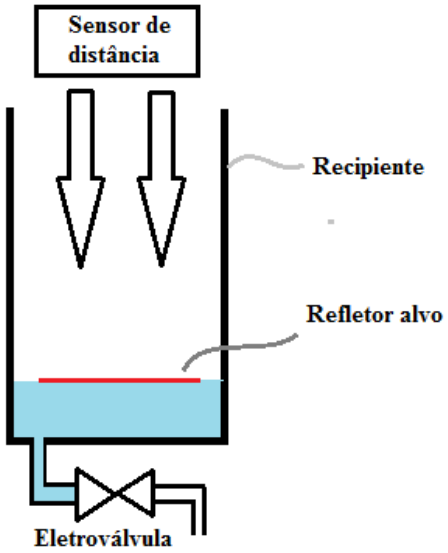



Figura 16 – Ilustração de funcionamento do pluviômetro com o sensor de distância

Para este projeto, a construção do pluviômetro é realizada utilizando o sensor ultrassônico juntamente com o material necessário. Alguns exemplos de sensores ultrassônicos foram pesquisados são apresentados na Figura 17 e suas características exibidas na Tabela 4.

(a) (b)

Figura 17 – Sensor de distância ultrassônico: (a) HC-SR04; (b) LV-EZ3.
Extraído de [36].

*Tabela 4 – Comparação de características dos sensores de precipitação.*

| | HC-SR04 | LV-EZ3 |
|---|---|---|
| **Faixa de Op.** | 2 a 400 cm | 1 a 400 cm |
| **Precisão** | 1 mm | 1 mm |
| **Resolução** | 0,3 mm | 0,3 mm |
| **Custo Médio** | R$ 25 | R$ 170 |

## 3.4. Escolha dos Sensores Meteorológicos

A escolha dos sensores foi feita após estudos dos principais instrumentos disponíveis no mercado. Os principais pontos que foram avaliados para sua escolha foram:

- Faixa de operação;
- Precisão;
- Resolução ou Sensibilidade;
- Corrente de funcionamento;
- Consumo energético;
- Relação custo benefício.

### *3.4.1. Sensor DHT22 - Higrômetro (Medidor de Umidade do Ar).*

Para medição da variável de Umidade, foi feita a escolha pelo sensor DHT22. A escolha deste sensor se deu principalmente pelo baixo consumo de energia deste componente se comparado aos outros sensores semelhantes estudados. Foi verificado que o sensor DHT22 apresenta características semelhantes e/ou iguais ao outros sensores existentes no mercado, apresentando, porém menor custo e melhor disponibilidade no mercado para compra do componente. Com a precisão de ± 2% e resolução de 0,1% o medidor atende perfeitamente as necessidades do projeto, além de consumir um baixo consumo de energia, conforme citado anteriormente.

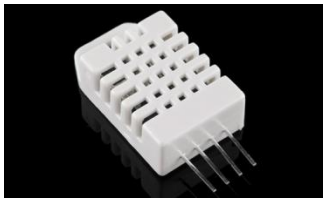


Figura 18 – Sensor de umidade do ar DHT22.
Extraído de [36].

Características do sensor:

- Entrada de 3,3 V a 6 V;
- Corrente de medição de 1 mA a 1,5 mA;
- Corrente de espera de 40 µA a 50 µA;
- Faixa de operação para umidade de 0 a 100% RH;
- Faixa de operação para temperatura de -40 ºC a 80 ºC;
- Interface com microcontrolador Arduino;
- Precisão de ± 2%;
- Resolução de 0,1%.

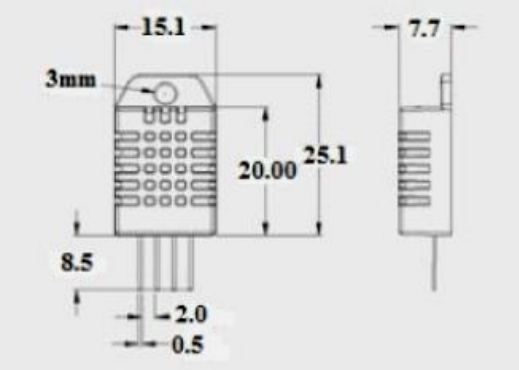



Figura 19 – Pinagem do sensor de umidade do ar DHT22.
Extraído de [36].

*Tabela 5 – Pinagem do sensor DHT22 a partir da esquerda pra direita.*

| Pino | Nome | Função |
|---|---|---|
| 1 | VCC | Alimentação |
| 2 | DATA | Transmissão dos Dados |
| 3 | - | Não Usado |
| 4 | GND | Terra |

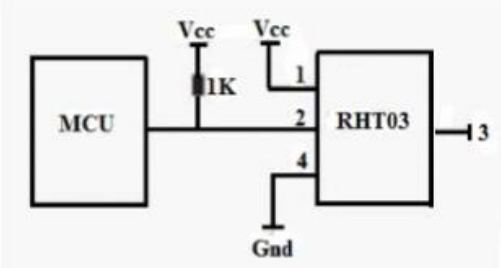



Figura 20 – Esquema elétrico do Sensor de umidade do ar DHT22.

Extraído de [18].

O princípio de funcionamento deste sensor é do tipo capacitivo.

Um sensor ou transdutor capacitivo funciona basicamente através deum capacitor que exibe uma variação do valor nominal da sua capacitância em função da variação de uma grandeza não elétrica. Uma vez que um capacitor consiste basicamente num conjunto de duas placas condutoras separadas por um dielétrico, as variações no valor nominal da capacidade podem ser provocadas por redução da área frente a frente, da separação entre as placas, ou por variação da constante dielétrica do material. Os sensores capacitivos permitem medir com grande precisão um grande número de grandezas físicas, tal como a umidade relativa do ar.

O funcionamento de um sensor capacitivo de umidade (designado sensor higrométrico) explora a dependência da constante dielétrica de alguns materiais com o teor de água no ar ambiente. O dielétrico é neste caso constituído por uma película fina de um material simultaneamente isolador e higroscópico o qual, dada a natureza porosa de um dos dielétricos, se encontra em contato com o ambiente cuja umidade relativa se pretende medir [11].

### *3.4.2. Sensor BMP180 – Barômetro (Medidor de Pressão Atmosférica) e Termômetro (Medidor de Temperatura).*

Para medição das variáveis citadas acima, foi feita a escolha pelo sensor BMP180, a escolha deste sensor se deu principalmente pela utilização de um único sensor para realização das duas medições necessárias. Além do óbvio ganho de espaço e de hardware, foi estudado que mesmo sendo utilizado para duas medições, o consumo deste componente é baixo comparado a utilização de dois sensores diferentes.

A escolha deste sensor foi feita também baseada na faixa de local de operação deste sensor, onde pode ser usado entre 0 e 9000 metros de altitude. O preço foi um dos fatores decisivos, onde

apresentou o melhor custo-benefício. A aplicação do projeto também foi fundamental para a escolha deste sensor, onde visto que estaremos sempre com uma pressão variando numa faixa bem pequena, visto, obviamente, que a embarcação estará sempre ao nível do mar.

Foi verificado que o sensor BMP180 apresenta características semelhantes e/ou iguais ao outros sensores existentes no mercado, apresentando, porém menor custo e a possibilidade, conforme citado anteriormente, da medição de duas variáveis em um único sensor.

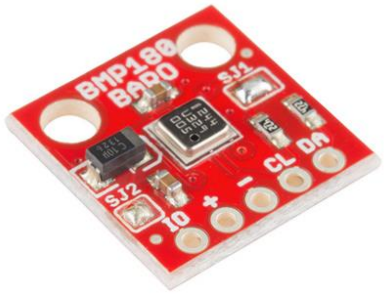



Figura 21 – Sensor de pressão atmosférica e temperatura BMP180.

Extraído de [36].

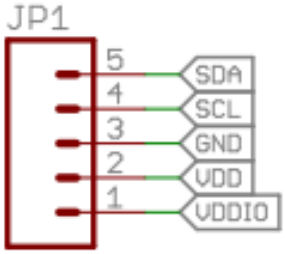



Figura 22 – Pinagem do sensor de pressão atmosférica e temperatura BMP-180.
Extraído de [17].

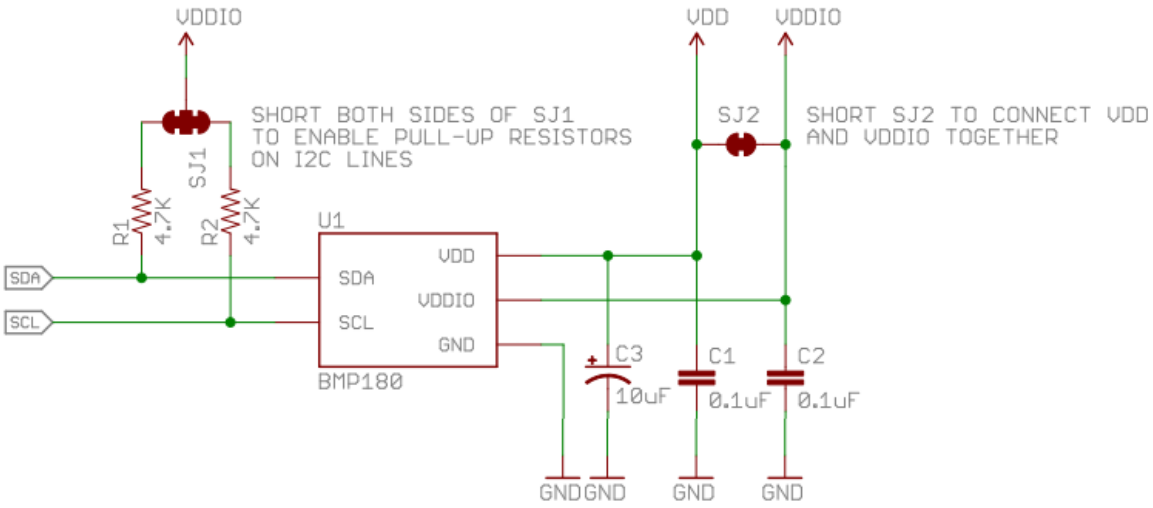



Figura 23 – Diagrama Elétrico do sensor de pressão atmosférica e temperatura BMP-180. Extraído de [17].

*Tabela 6 – Pinagem do sensor de pressão atmosférica e temperatura BMP-180*

| Pino | Nome | Função |
|---|---|---|
| 1 | VDDIO | Alimentação Digital (Não utilizada) |
| 2 | VDD | Alimentação |
| 3 | GND | Terra |
| 4 | SCL | Entrada de *Clock* Serial |
| 5 | SDA | Entrada de Dados Serial |

Características do sensor:

- Alimentação de 1,8 V a 3,6 V;
- Corrente de medição: 5 µA;
- Corrente de espera: 0,1 µA;
- Faixa de operação de 300 hPa a 1100 hPa para pressão atmosférica;
- Faixa de operação de -40 ºC a 80 °C para temperatura;
- Tempo de reação: 7,5 ms em média;
- Dimensões: 14 x 12 mm;
- Peso: 1,2g;
- Precisão do sensor barométrico: ± 0,12 hPa;
- Precisão do sensor térmico: ± 0,5 °C;
- Resolução do sensor barométrico: 0,01 hPa;
- Resolução do sensor térmico: 0,1°C;

O princípio de funcionamento deste sensor é do tipo piezo resistivo.

Um material piezelétrico é aquele que, quando exercido uma força sobre o material este traduz a força em diferença de potencial e vice versa. Um transdutor piezoresistivo funciona da mesma forma, ou seja, reverte a força exercida sobre ele em uma diferença de resistência.

Quando ocorrem variações da pressão, ocorrem alterações nas resistências implantadas de acordo com o efeito piezo resistivo. Esta resistência é alimentada por uma carga parte de um sistema de controle o qual pode ser analisado. A medição é obtida a partir deste sistema de controle o qual traduz para uma saída de 4 a 20 mV que é convertida em um sinal digital por um conversor A/D interno e enviada para o microcontrolador [12].

### *3.4.3. Sensor HC-SR04 usado no pluviômetro (Medidor de quantidade de Precipitação)*

De acordo com os projetos estudados na seção 3.3.4, para medição da variável pluviométrica utilizando o sensor de distância ultrassônico, foi feita a escolha pelo sensor HC-SR04. Sua faixa de operação é de 2 centimetros até 4 metros de distância do alvo, sendo perfeitamente aceitável no nosso projeto. Com uma precisão de 1 milimetro, este sensor atende o principio que foi estudado e será utilizado. A escolha deste sensor foi pensada também na sua aplicação no projeto, na construção do pluviômetro, onde este sensor, juntamente com o recipiente para medição, foi objetivado para ocupar o menor espaço possível para mantermos a idéia de uma Estação Meteorológica remota de fácil instalação e funcionamento.

Foi verificado que o sensor HC-SR04 apresenta características semelhantes e/ou iguais ao outros sensores existentes no mercado, apresentando, porém menor custo, alta disponibilidade no mercado e utilização simplificada foram decisivos para a sua escolha.

O princípio de funcionamento de um sensor ultrassônico é o mesmo de um sonar, onde o seu funcionameto é baseado na emissão de uma onda sonora de alta freqüência, e na medição do tempo levado para a recepção do eco produzido quando a onda se choca com um objeto capaz de refletir o som. O transmissor emite pulsos ultrassônicos ciclicamente. Quando um objeto reflete este pulso, o eco é percebido pelo receptor e convertido em sinal elétrico internamente.

Estes sensores funcionam medindo o tempo de propagação do eco, ou seja, medindo o intervalo de tempo entre o impulso emitido e o eco refletido do mesmo. Este intervalo medido é feito através da velocidade do som no ar, que é de 340 m/s, e de forma mais simplificada, a distancia em centímetros do objeto é dada pelo intervalo em microssegundos entre emissão e recepçao divididos por 58 [13].


Figura 24 – Sensor de distância ultrassônico HC-SR04.
Extraído de [36].

O seu diagrama esquemático está no Anexo A.

Características do sensor:

- Alimentação de 5 V;
- Corrente de operação: 15 mA;
- Corrente de espera: 2 mA;
- Faixa de operação de 2 cm a 4 metros;
- Precisão de 1 milimetro;
- Tempo de reação: 7,5 ms em média;
- Dimensões: 45 x 20 x 15 mm.

*Tabela 7 – Pinagem do sensor de distância ultrassônico HC-SR04.*

| Nome | Função |
|---|---|
| VCC | Alimentação |
| Trigger | Entrada do sinal de transmissão |
| Echo | Saída do sinal de eco (distância) |
| GND | Ground |

### 3.5. Estudo dos tipos de Comunicação

Foram avaliadas diversas características dos tipos de comunicação estudados, como a perda na transmissão dos dados e imunidade ao ruído, para que não haja erros de comunicação entre transmissor e recepção dos dados transmitidos.

Em relação à banda de transferência de informações entre a central de comando e o terminal receptor de dados, neste projeto não existe a necessidade de uma alta velocidade, pois não será critico que o sistema receba informação dos dados instantaneamente, porque nas possíveis aplicações do projeto, este não deverá apresentar variações dos valores que necessitem estudo imediato do acontecimento.

### *3.5.1. Zigbee*

O protocolo de comunicação Zigbee é baseado no padrão IEEE 802.15.4 para áreas de redes pessoais, operando na frequência ISM (*Industrial Scientific and Medical*), onde na Europa é de 868 MHz (1 canal), nos Estados Unidos é 915 MHz (10 canais) e em outras partes do mundo é 2,4 GHz(16 canais) e não requer uma licença para o funcionamento. Outra característica desse tipo de comunicação é a imunidade a interferências e tem capacidade de hospedar milhares de dispositivos numa rede, aproximadamente 65000 por canal, com taxas de transferências de dados que variam de 20 kbps a 250 kbps.

O Zigbee possui um baixo consumo de energia elétrica, pois o módulo Zigbee quando não está transmitindo ou recebendo dados entra em um estado de latência, consumindo pouca energia e esse transceptor, em especial possui um alcance de rádio frequência em visada direta para ambientes externos de até 1,6 km [14].

### *3.5.2. RF de 433 MHz*

O módulo RF de 433 MHz é composto de componentes básicos para a comunicação em rádio frequência. É popularmente usado em controles remotos, sistemas de alarmes e robótica em geral. O caso mais usado é para transmissão de dados em baixa velocidade e curta distância, onde, por exemplo, poderá enviar dados de um sensor, ou controlar o acionamento de um dispositivo.

Pode ser ligado diretamente ao microcontrolador, por exemplo, no Arduino. A comunicação é unidirecional, ou seja, apenas o transmissor envia os dados para o receptor.

Em condições ideais, pode chegar ao alcance de 200 metros, mas normalmente a distância é inferior, em uma faixa de 20 a 150 metros. Para melhorar o alcance, pode-se soldar um fio de cobre no furo próprio para antena [14].

### *3.5.3. Módulo Rádio Frequência Apc220*

Este módulo pode ser uma solução para quem precisa transmitir dados para o microcontrolador, por exemplo, o Arduino, que esteja agrandes distâncias, porque o seu alcance característico é de aproximadamente 1000 metros em média, podendo alcançar 1200 metros em área aberta. Por formar uma comunicação ponto-a-ponto, não depende de infraestrutura de rede, cabeamento ou outros equipamentos para funcionar. Possui modulação GFSK (*Gaussian frequency-shift keying*), que consiste basicamente em atribuir frequencias diferentes em relação ao bit que é transmitido. Ou seja, quando um bit 0 é transmitido, a portadora assume uma frequencia correspondente durante a duração deste bit, de maneira análoga é feita a transmissão do bit 1, assumindo um segundo valor de frequência. Antes de entrarem no modulador, passam por um filtro gaussiano de modo a reduzir a largura espectral e assim suavizar a transição dos valores dos pulsos. A modulação GFSK é utilizada nos sistemas *Bluetooth*, uma vez que provê uma melhor eficiência espectral em relação à modulação FSK [15].

A frequência de operação pode ser ajustada via *software* (418 a 455 MHz), e a taxa de transferência pode chegar a 19,2 kbps. Para funcionar precisa de um *software* chamado *RF-Magic*, que configura as características do módulo, como frequência, o NET ID e as configurações da serial, como velocidade e paridade.

### *3.5.4. WLAN*

Uma WLAN (*Wireless Local Area Network*) é uma rede local sem fio, implementada como extensão ou alternativa para redes convencionais cabeadas. Além de disponibilizar a implementação de redes locais, esta tecnologia pode ser utilizada para o acesso à internet. As WLANs mais comuns são as redes Wi-Fi (*Wireless Fidelity*) e se refere qualquer tipo de rede que utiliza o padrão IEEE 802.11.

O padrão IEEE 802.11 possibilita a transmissão de dados em diferentes taxas, dependendo da versão empregada, e específica uma arquitetura comum, métodos de transmissão, e outros aspectos de transferência de dados sem fio, permitindo a interoperabilidade entre os diversos produtos WLAN.

WLANs fornecem conexões de rede por meio de ondas de rádio frequência, as faixas de frequência de ondas de rádios utilizadas nessas redes são especificadas pela ITU (*International Telecommunication Union*). As faixas de frequências de 2,4 GHz e 5,0 GHz são faixas de freqüência que não precisam ser licenciadas para uso de comunicações, por isso são usadas para WLAN [14].

### 3.6. Escolha da Comunicação

A comunicação da estação com o computador local será feita através do Módulo Rádio *Wireless* Apc220.

Essa forma de comunicação atende perfeitamente as necessidades do projeto, pois o mesmo consegue transmitir de forma eficiente a longas distâncias, tendo alcance do sinal de até 1200 metros. Foi analizada também a praticidade oferecida, por não depender de infraestrutura de rede, cabeamento ou outros equipamentos para funcionar, ao contrário de todos os componentes estudados a realização deste item. Sua configuração via *software* foi o diferencial para sua escolha, além de sua disponibilidade no mercado, mesmo apresentando um custo acima da faixa do mercado.

A frequência de operação pode ser ajustada via *software* (418 a 455 MHz) e a taxa de transferência pode chegar a 19,2 kbps.

Características do sensor:

- Faixa de Operação variavel entre 37 MHz;
- Taxa de transferência variavel de 1,2 kbps a 19,2 kbps;
- Corrente de operação: entre 25 mA e 35 mA;
- Alcance máximo: até 1200m em área aberta;
- Dimensões: 39 milímetros x 19 milímetros x 2,8 milímetros.

Figura 25 – Conjunto do Módulo Rádio *Wireless* Apc220.
Extraído de [36].

*Tabela 8 – Pinagem do Módulo Apc220.*

| Pino | Nome | Função |
|---|---|---|
| **1** | GND | Ground |
| **2** | VCC | Alimentação DC - 3.5V-5.5V |
| **3** | EN | Power Enable (Não utilizado) |
| **4** | RXD | UART input, TTL |
| **5** | TXD | UART output, TTL |
| **6** | MUX | Outras funções (Não utilizado) |
| **7** | SET | Definição de Parâmetros (Não utilizado) |

# 4. PROJETO DO PROTÓTIPO

## 4.1. Projeto do Circuito Eletrônico

No capítulo 3, foi realizado o estudo sobre a seleção de quais componentes que vão atender as variáveis necessárias para a criação da Estação Meteorológica, a escolha dos sensores que atendem estas variáveis e, conseqüentemente, o circuito eletrônico individual do microcontrolador, dos sensores e do módulo de comunicação. Após esse estudo veremos, neste capítulo, a integração de todos os componentes para formar o circuito eletrônico final de toda a estação.

Conforme vimos no capitulo 3, o microcontrolador Arduino Uno possui 14 terminais digitais, 5 terminais analógicos e outros 8 terminais de tensão. Mostraremos a seguir, nas próximas subseções, como cada componente se ligará ao microcontrolador, especificando quais são os terminais de dados e os de alimentação, formando ao final de todo o processo, que será mostrado, a inteligação de todos os componentes.

Todo este processo será mostrado através de imagens do Fritizing, um *software* auxiliar utilizado para exibição das interligações [16].

A arquitetura do hardware, apresentado em diagrama de blocos é mostrado na Figura 26 abaixo.

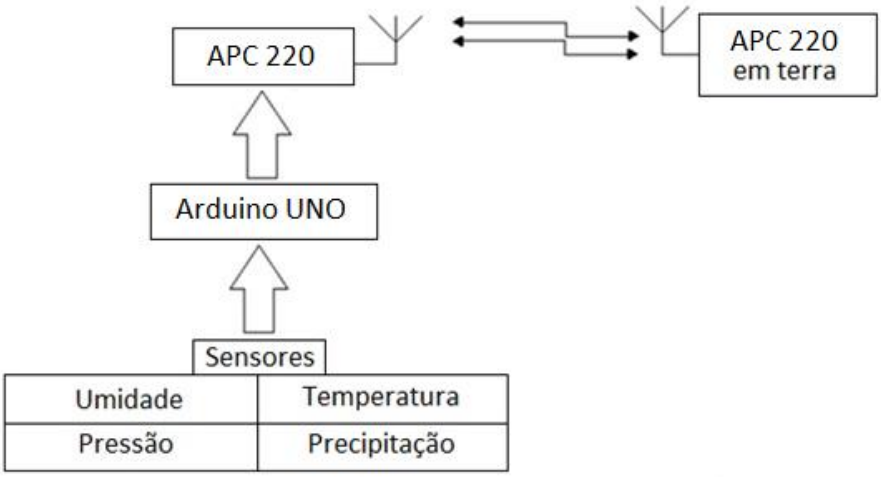



Figura 26 – Diagrama de Blocos da Estação Meteorológica.
Fonte: Autor.

*4.1.1. Sensor de Temperatura e Pressão BMP180*

De acordo com o estudado no capítulo anterior, o módulo BMP180 possui interface I2C para aquisição de dados, através dos pinos SDL e SDA do sensor, não precisando de componentes externos para operar [17]. Assim, as ligações deste sensor com o microcontrolador é realizado de maneira direta. A interligação elétrica necessária para funcionamento é mostrada na Figura 27.

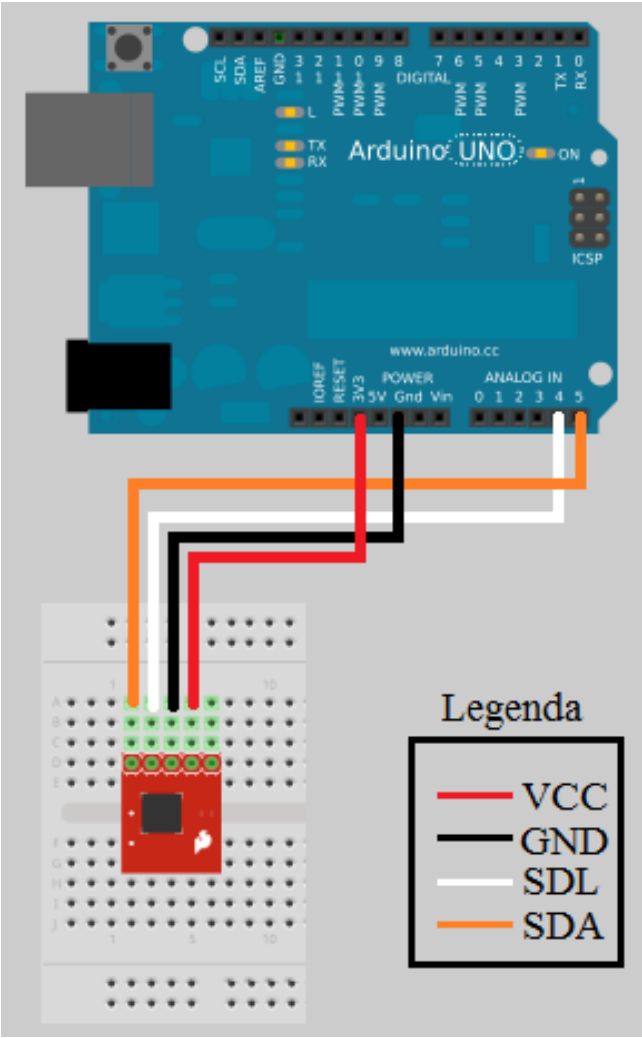



Figura 27 – Interligação do sensor BMP180 com o Arduino realizado no *Fritzing*.
Fonte: Autor.

### *4.1.2. Sensor de Umidade DHT22*

De acordo com o estudado no capítulo anterior, o sensor DHT22 coleta os dados através do protocolo 1-Wire, retornado o valor aquisitado entre as diferenças dos bulbos digitalmente através do pino 3 do sensor. O resistor de *pull-up*, entre o terminal VCC e o de Dados, de 1 kΩ foi adicionado para o funcionamento do sensor conforme sugerido pelo fabricante [18].

A interligação elétrica necessária para funcionamento é mostrada na Figura 28.

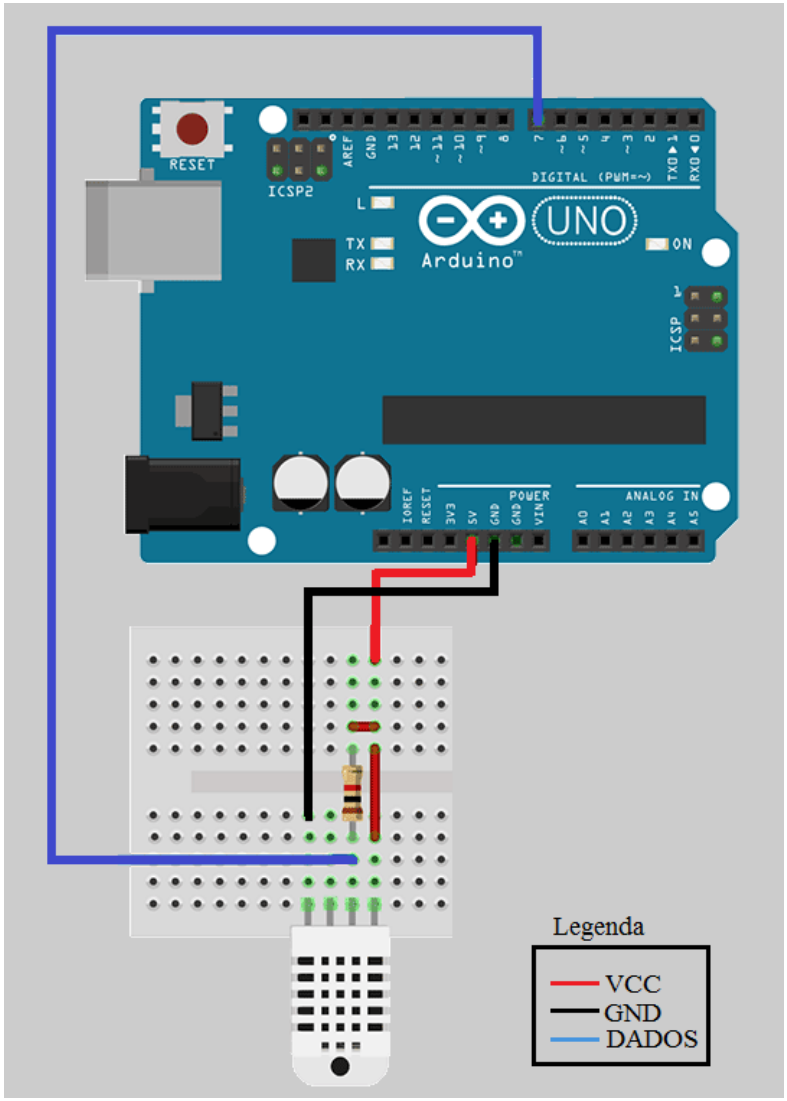



Figura 28 – Interligação do sensor DHT22 com o Arduino realizado no *Fritzing*.
Fonte: Autor.

*4.1.3. Sensor de Distância HC-SR04*

O sensor de distância coleta a informação da distância entre o refletor alvo e o sensor através da multiplicação da velocidade do som vezes o tempo. Esta multiplicação é dividida por 2 porque é a onda sonora percorre essa distância 2 vezes, uma pela onda emitida e outra pela onda refletida [19]. De acordo com o *aplitcation note* do sensor, nenhum componente elétrico/eletrônico externo é necessário para seu funciomento. A interligação elétrica necessária para funcionamento é mostrada na Figura 29.

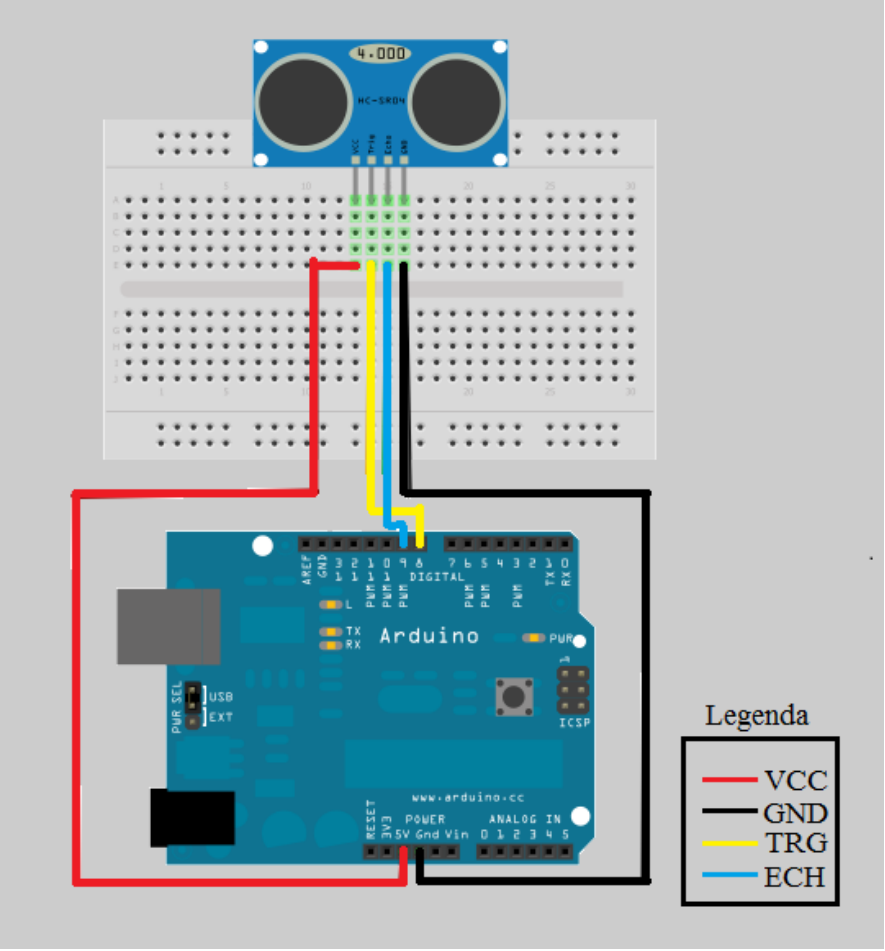



Figura 29 – Interligação do sensor HC-SR04 com o Arduino realizado no *Fritzing*.
Fonte: Autor.

*4.1.4. Módulo de Comunicação APC220*

O módulo de RF APC 220 necessita de apenas quatro dos seus sete terminais para funcionamento, onde os dois são de alimentação e o par para transmissão e recepção de dados.

Os pinos digitais 0 e 1 são utilizados, por padrão do microcontrolador, para a comunicação serial direta, através da porta COM, entre o microcontrolador e computador. Então foi utilizada uma biblioteca para a criação de utilização de dois outros pinos digitais disponíveis no módulo para o uso de transmissão e recepção do fluxo de dados.

O módulo também não exige componentes externos para o seu funcionamento, sendo feita assim, ligações diretas entre o módulo e o microcontrolador [20]. A interligação elétrica necessária para funcionamento é mostrada na Figura 30.

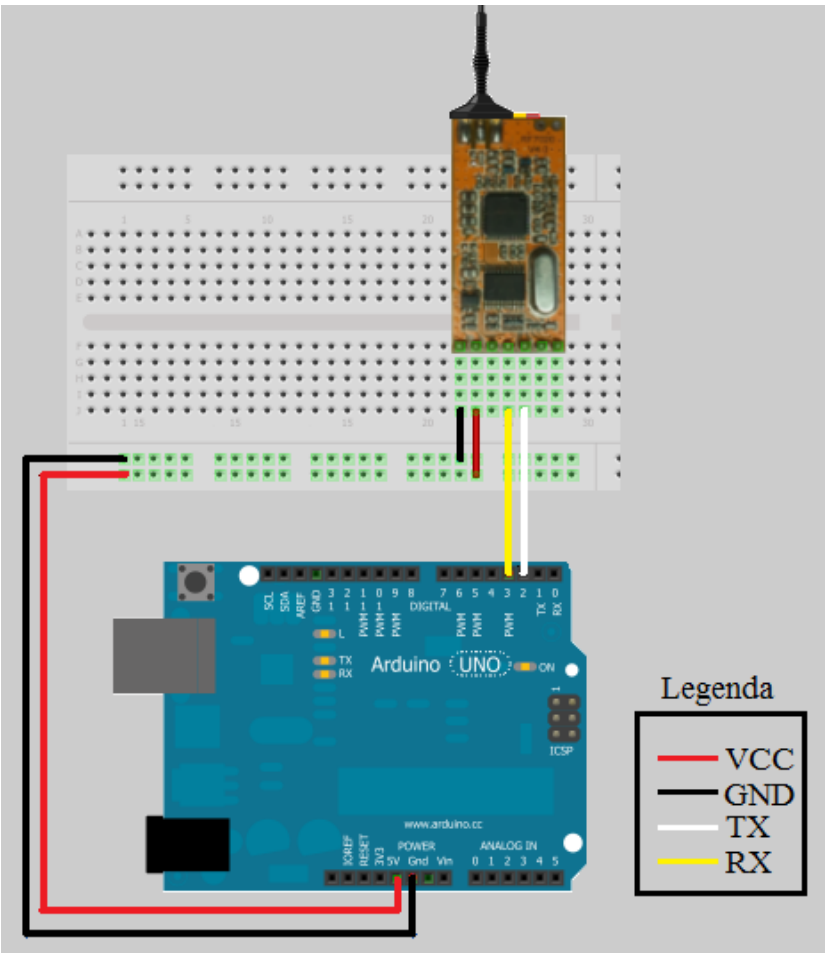



Figura 30 – Interligação do módulo APC220 com o Arduino realizado no *Fritzing*.

Fonte: Autor.

*4.1.5. Estação Meteorológica*

A Figura 31 apresenta a interligação entre todos os componentes usados na construção do protótipo.

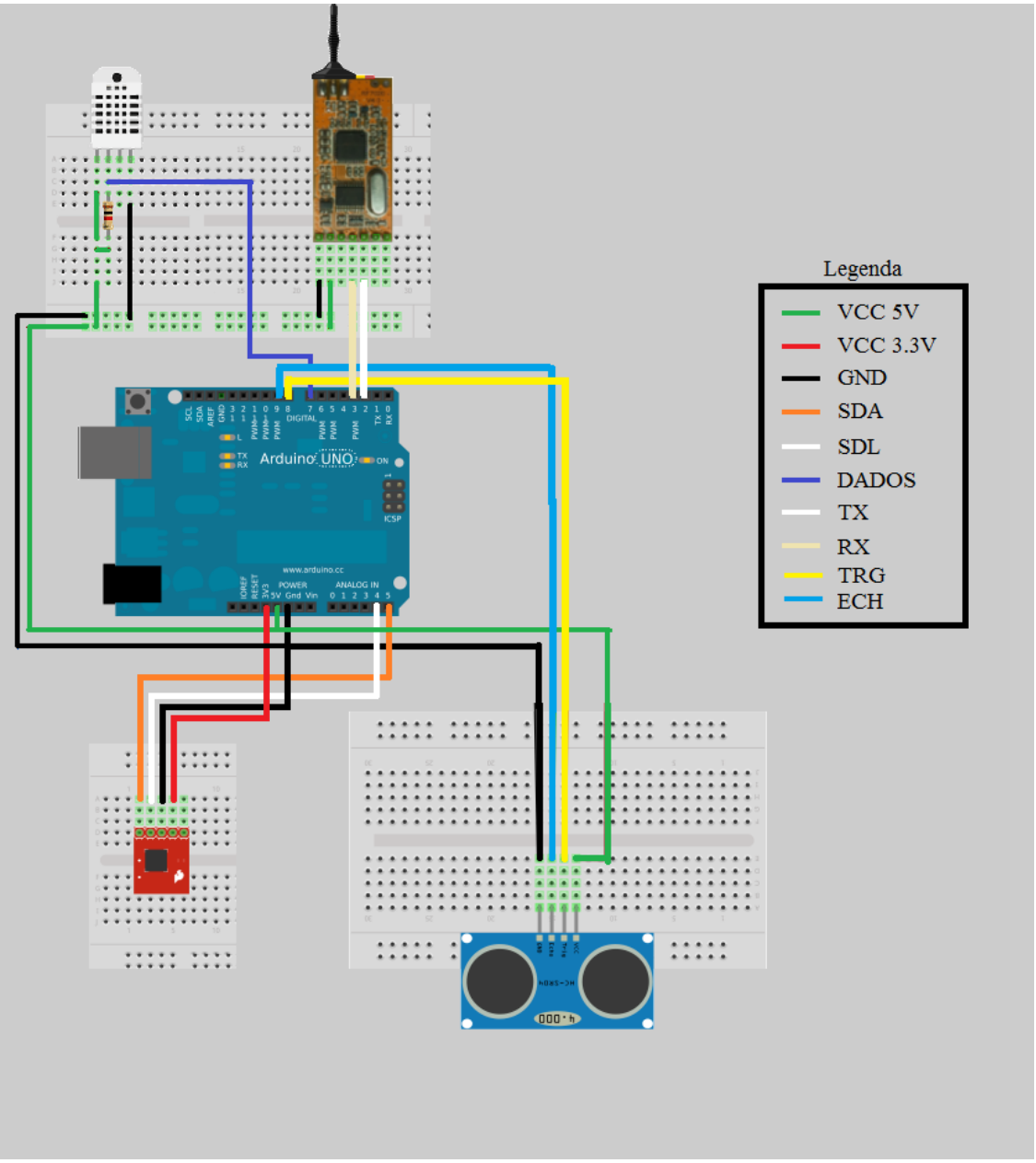



Figura 31 – Interligação completa da Estação Meteorológicarealizado no *Fritzing*.
Fonte: Autor.

## 4.2. Construção da Estação Meteorológica

De acordo com os circuitos eletrônicos da Estação Meteorológica visto na seção 4.1, iniciamos a construção do protótipo com a montagem dos circuitos dos sensores. As cores dos fios de interligação vistas nas imagens do projeto teórico foram utilizadas para ser igual ao da construção real do protótipo conforme veremos ao longo desta seção.

Para a construção da estação, foi necessária a construção e utilização de um barramento de alimentação utilizado por todos os componentes. Esta placa, que está devidamente identificada, está localizada ao centro do protótipo. Foram utilizados também materiais auxiliares como terminais macho e fêmea para interligação, soquetes de circuitos integrados e cola de silicone.

### *4.2.1 Sensor de Temperatura e Pressão BMP180*

Conforme mostrado nas Figuras 30 e 31, a placa do sensor BMP180, responsável pela aquisição das medidas de temperatura e pressão, não exige nenhum componente eletrônico externo para seu funcionamento. Então, com o auxílio de terminais, de um soquete de circuito integrado e de uma placa de padrão sem conexões, foi realizada a interligação entre os terminais do módulo e os pinos do microcontrolador. A Figura 34 apresenta a ligação real do sensor para colocá-lo em funcionamento.

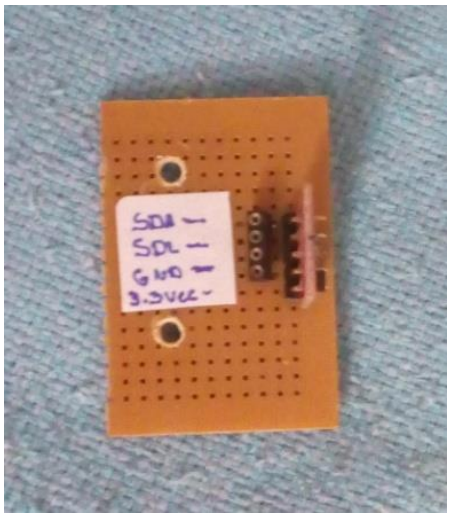



Figura 32 – Vista Superior do Sensor de temperatura e pressão BMP180.
Fonte: Autor.

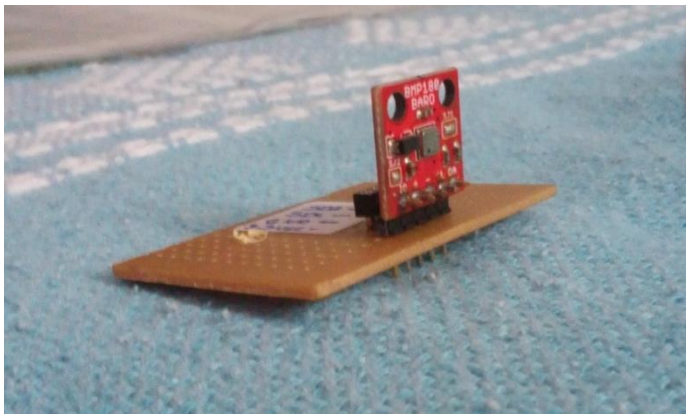


Figura 33 – Vista Frontal do Sensor de temperatura e pressão BMP180.
Fonte: Autor.

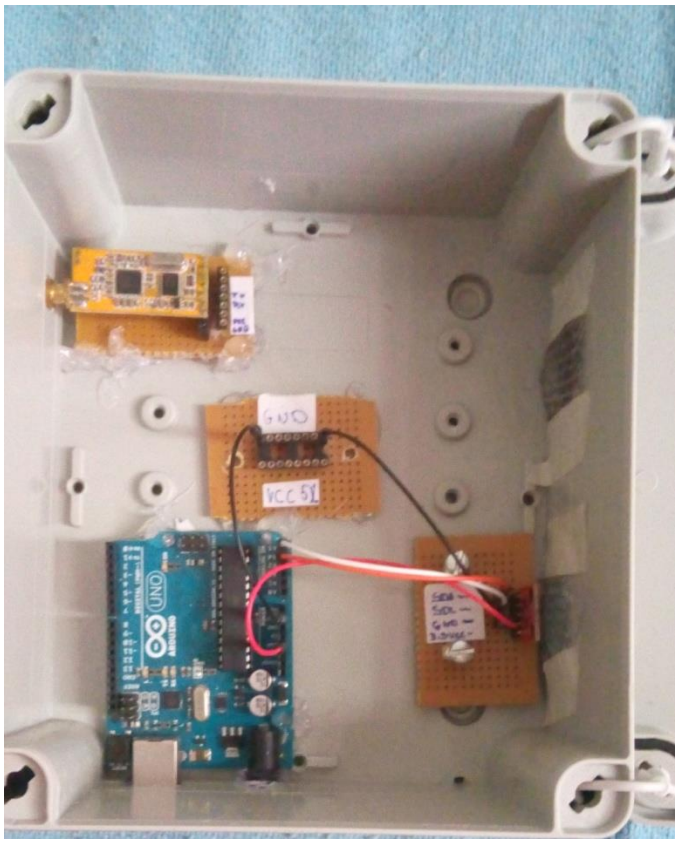


Figura 34 – Interligação do Sensor de temperatura e pressão BMP180 com o Arduino.
Fonte: Autor.

*4.2.2. Sensor de Umidade DHT22*

O sensor DHT22, responsável pela umidade, necessita de um resistor de *pull-up* sugerida pelo fornecedor para seu funcionamento. Buscando referencias no *datasheet* e conforme calibração, que veremos na seção 6.2, escolheu-se o resistor 1 kΩ para contemplarmos o funcionamento desde módulo. Contou-se novamente com o auxílio de terminais e de uma placa padrão para interligação entre o sensor, o resistor e o microcontrolador. A Figura 37 apresenta a ligação real do sensor para colocá-lo em funcionamento.


Figura 35 – Vista Superior do Sensor de umidade DHT22.
Fonte: Autor.

Figura 36 – Vista Frontal do Sensor de umidade DHT22.
Fonte: Autor.

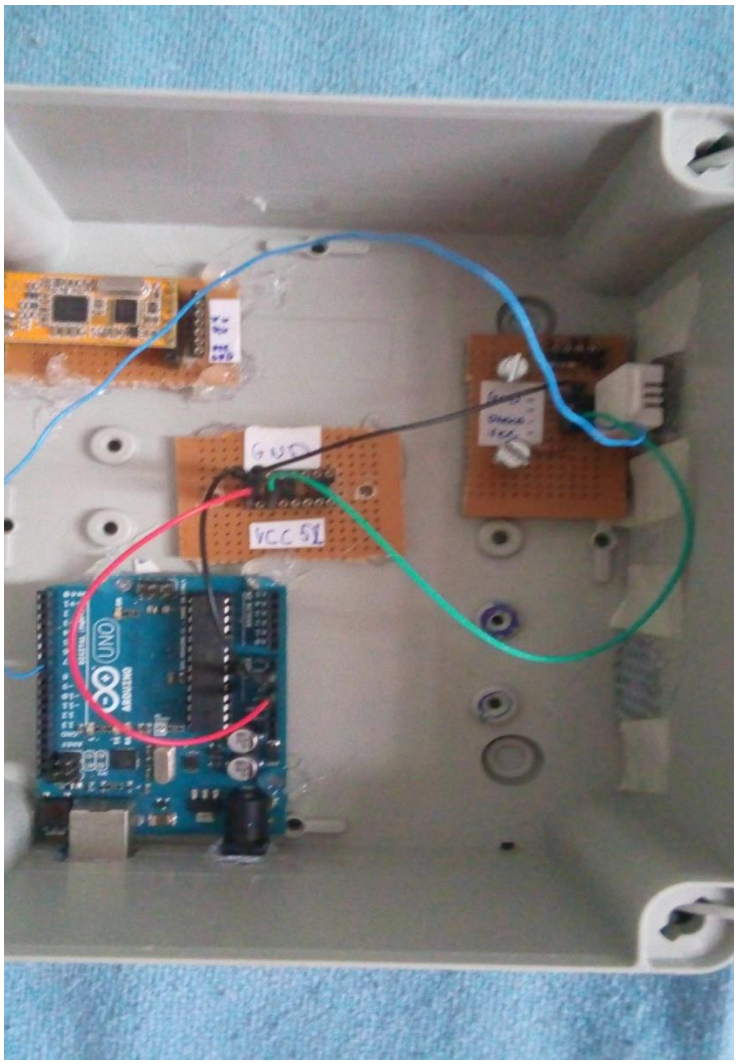



Figura 37 – Interligação do Sensor de umidade DHT22 com o Arduino.
Fonte: Autor.

*4.2.3. Sensor de Distância HC-SR04*

O sensor HC-SR04, responsável pela medição de precipitação, não exige nenhum componente eletrônico externo para seu funcionamento. Na sua interligação com o microcontrolador foi utilizado terminais de contatos. A Figura 40 apresenta a ligação real do sensor para colocá-lo em funcionamento.

Figura 38 – Vista superior do Sensor de distância HC-SR04 instalado no pluviômetro.
Fonte: Autor.

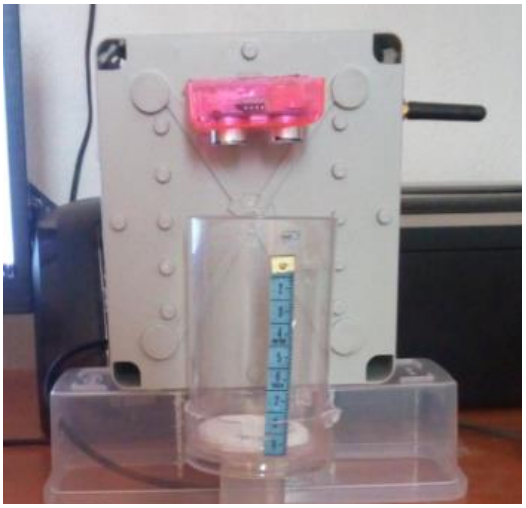


Figura 39 – Vista frontal do Sensor de distância HC-SR04 instalado no pluviômetro.
Fonte: Autor.

Figura 40 – Interligação do Sensor HC-SR04 com o Arduino instalado no pluviômetro.
Fonte: Autor.

*4.2.4. Módulo de Comunicação APC220*

O módulo de comunicação APC220, responsável pela comunicação em RF entre a estação e computador, não exige nenhum componente eletrônico externo para seu funcionamento. Novamente foram utilizados os terminais auxiliares. A Figura 43 apresenta a ligação real da placa de comunicação para colocá-lo em funcionamento.

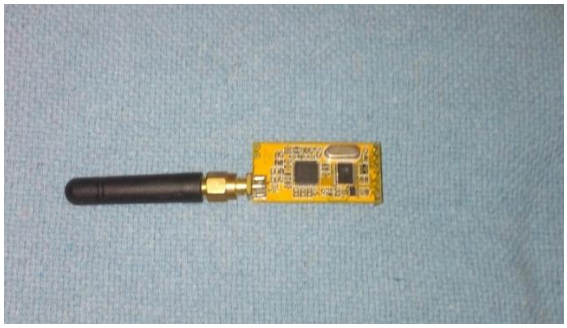


Figura 41 – Vista Superior do módulo de comunicação APC220.
Fonte: Autor.

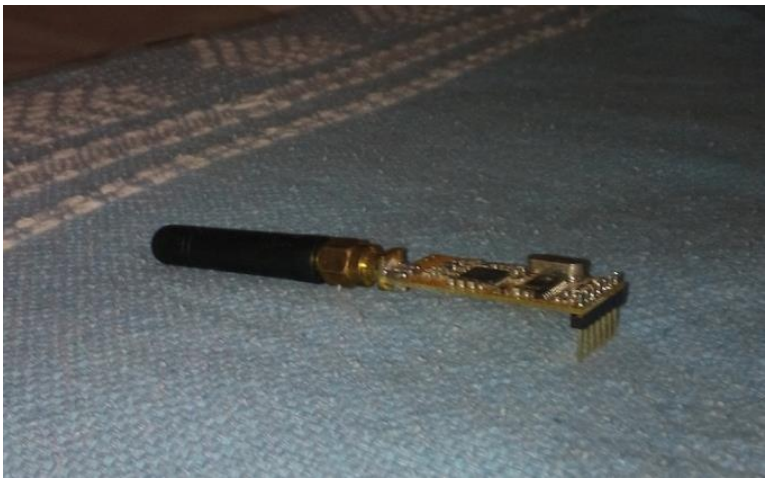


Figura 42 – Vista Frontal do módulo de comunicação APC220.
Fonte: Autor.

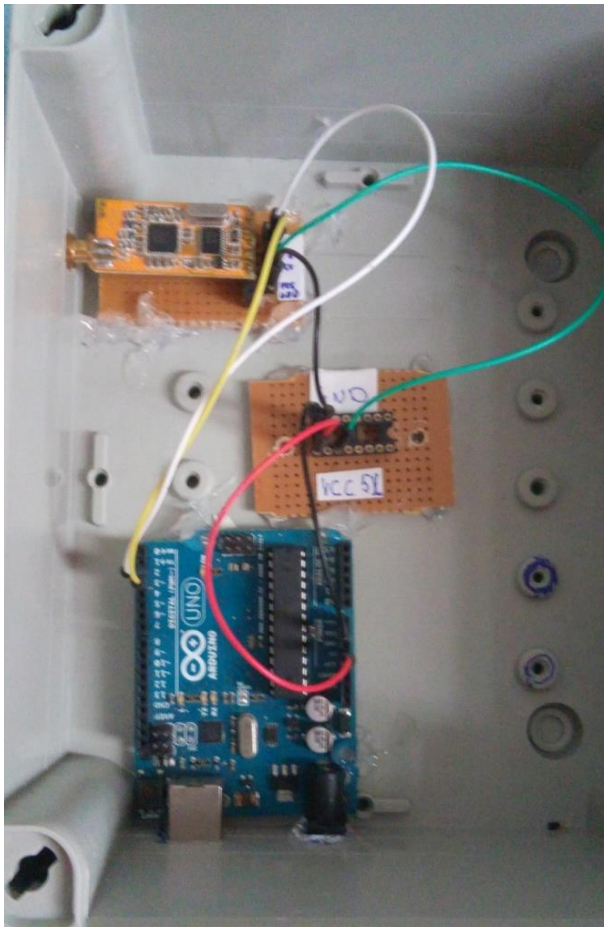



Figura 43 – Interligação do módulo de comunicação APC220 com o Arduino.
Fonte: Autor.

### 4.3. Caixa Estanque

Depois de alguns estudos feitos sobre todo o projeto relacionado a Embarcações Teleoperadas para Sistema de Monitoramento e Defesa, é necessária para a construção da Estação Meteorológica e recepção dos componentes eletrônicos, uma caixa a prova d'gua.

Para este item, foi escolhida uma caixa de dericação eletrica Caixa Light da Steck, com duplo isolamento, sem elementos metalicos e com dobradiça articulada [21]. Sua aparência e dimensões são exibidas nas Figuras 44 e 45.

Figura 44 – Caixa de derivação Light Steck.
Extraído de [21].

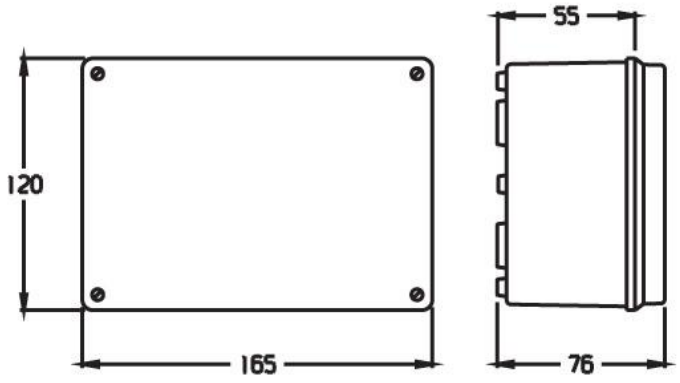
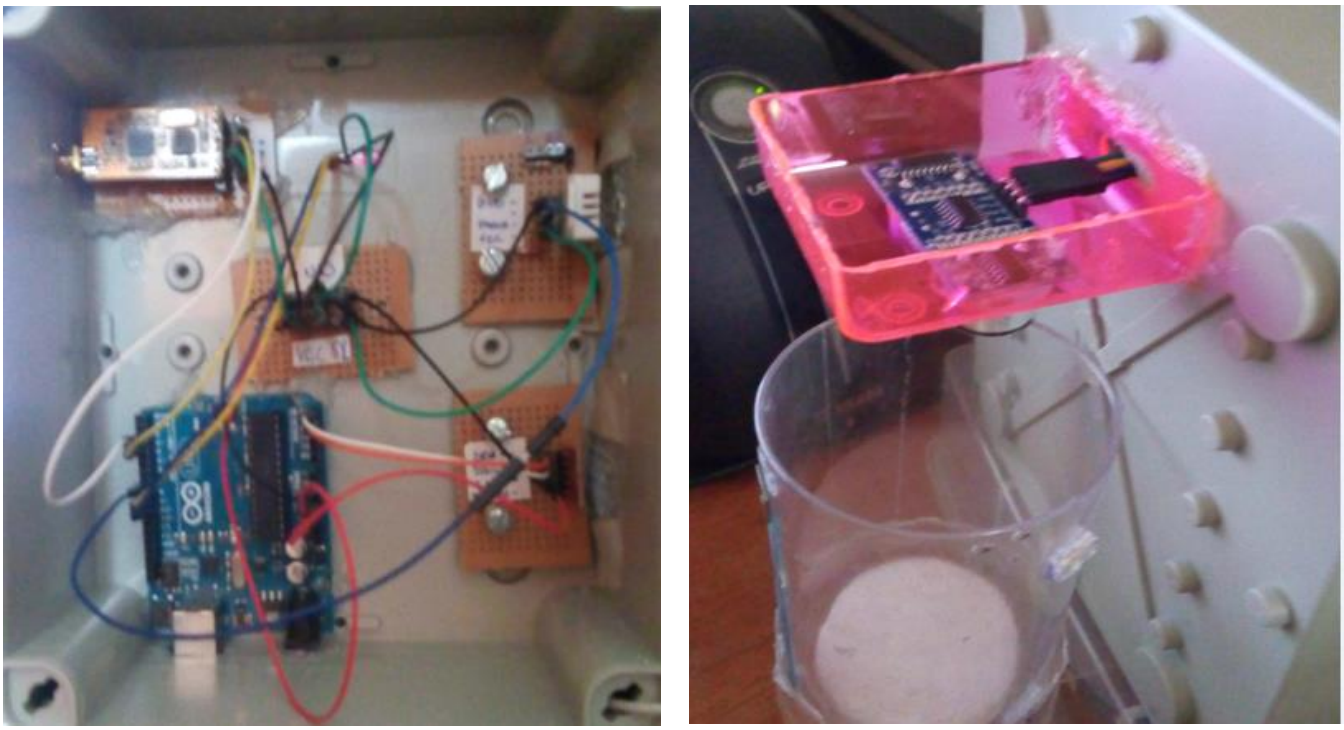



Figura 45 – Dimensões da caixa de derivação Light Steck.
Extraído de [21].

Para a aquisição dos dados, transmissão do sinal e alimentação do circuito, foram feitas aberturas nas laterais da caixa, tornando-a assim, vulnerável à presença de água novamente. Para reforçar a proteção que foi vulnerabilizada pelas janelas, deve ser projetada uma segunda caixa, de um material a prova de penetração de água, como o acrílico, maior que a caixa de derivação, com um apoio para um mastro, de forma a tornar a estação sustentável verticalmente, já que todas as laterais possuem conexões com o ambiente externo, impossibilitando assim o apoio em um desses lados.

### 4.4. Protótipo da Estação Meteorológica

Ao longo do capítulo foram apresentados todos componentes necessários para a construção da estação. Nesta seção veremos a aparência de final do protótipo já montado e funcionando.



(a) (b)

Figura 46 – (a) e (b) Vistas da interligação completa da Estação Meteorológica.
Fonte: Autor.

# 5. *SOFTWARE* DE AQUISIÇÃO DE DADOS METEOROLÓGICOS

Este capítulo tratará de toda a interface de *software* necessária para a realização do projeto, comentando da instalação e execução do Arduino, do *Microsoft Visual Basic* para realizar a aquisição das variáveis mensuradas através dos sensores. Todo o projeto foi pensado para organização, transmissão e exibição dos dados de forma simples e eficiente.

## 5.1. Ambiente de Desenvolvimento – Arduino

Como visto no Capitulo 3, o Arduino é um microcontrolador de hardware bastante definido com um *software* bastante intuitivo e de fácil utilização. Este microcontrolador pode receber como entrada uma variedade de sensores, motores DC, atuadores entre outros. Ao receber estas informações, pode ser configurado a tratar esses dados de maneira que atenda de forma interativa ao projeto que foi planejado. O microcontrolador é programado usando a linguagem de programação Arduino e o ambiente de desenvolvimento Arduino, sendo todo este *software* de programação de código-livre [6].

### *5.1.1. Instalando o software Arduino*

Para instalar o Arduino no computador, faz-se necessário baixar o instalador com o pacote de drivers existente no site oficial. Os passos, disponíveis no site oficial, são os seguintes:

1. Conecte o Arduino e espere o Windows começar o processo de instalação.

2. Após alguns momentos o processo irá falhar.

3. Clique em menu iniciar e abra o painel de controle.

4. No painel de controle, acesso item Sistema e logo após clique em Gerenciador de dispositivos.

5. Procure dentre os itens Portas de Comunicação (COM & LPT) um item chamado Arduino UNO (COMxx).

6. Clique com o botão direito nesse item e escolha “Atualizar Driver”.

7. Escolha a opção “Navegar no meu computador em busca dos drivers”.

8. Aponte para a subpasta Drivers e escolha o arquivo "ArduinoUNO.inf".

9. O *Windows* terminará a instalação dos drivers.

*5.1.2. Executando o softwarede interface de desenvolvimento*

Para a utilização do *software* não é necessária mais nenhuma instalação, sendo apenas requisitada a descompactação do arquivo baixado diretamente site oficial feito anteriormente e executar o programa “Arduino.exe”, apresentada na Figura 47.

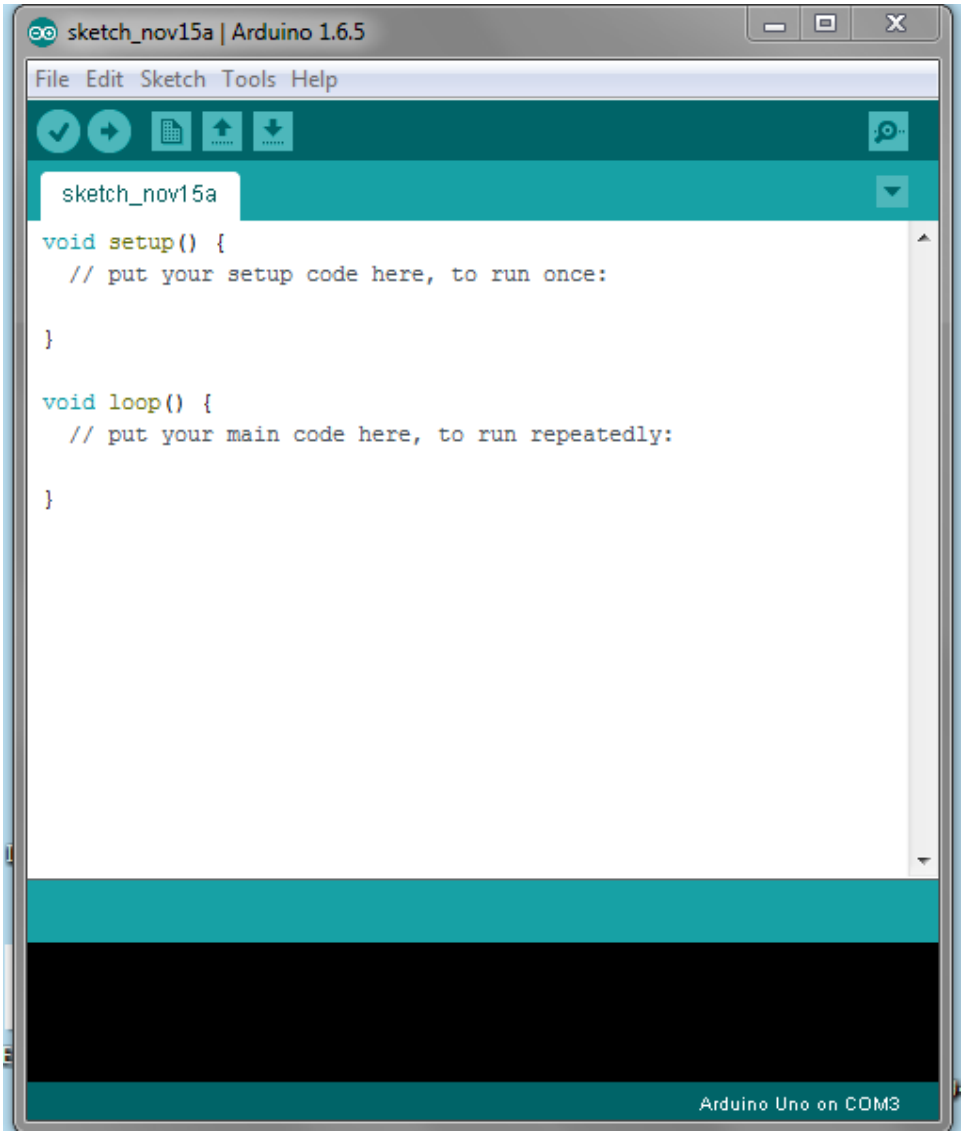



Figura 47 – Ambiente de desenvolvimento Arduino.
Fonte: Autor.

*5.1.3. Código Implementado*

Este código inicialmente testa os sensores, trata da aquisição dos dados pelos sensores e transmissão, de modo simplificado pelo módulo de comunicação APC220 que está na embarcação remota, enviando os dados de modo organizado, para melhor tratamento dos dados pelo *software* de monitoramento localizado no computador em terra.

Os sensores necessitam de biblioteca para seu funcionamento. Estas foram baixadas diretamente do site *Sparkfun*, responsável pela fabricação do sensor de pressão e temperatura BMP180, do site do Arduinopara o sensor DHT22 e para o sensor HC-SR04.

A plataforma Arduino é o responsável pelo teste dos sensores, aquisição de dados, que lerá os valores dos dados fornecidos pelos sensores e enviá-los através da porta serial e através do módulo de comunicação, tornando assim possível, a leitura por dois canais diferentes. Inicialmente é definida como 9600 bps como a velocidade de comunicação da porta serial, por que é o valor que a comunicação em RF tem maior alcance. Em seguida, são feitos os testes dos sensores para checarmos se estão em condição de funcionamento. Caso algum destes testes responda negativamente, é necessária a avaliação do erro encontrado. Se a condição de testes for satisfeita, inicia-se o processo de aquisição, com o Arduino lendo os dados de todos os sensores e enviando-os através da porta COM, podendo ser exibidos no próprio *software*, através da tela Serial Monitor.

Inicialmente, o procedimento de teste dos sensores é realizado de acordo com os exemplos fornecidos em seus respectivos *datasheet*, não existindo maiores informações do procedimento interno dos mesmos, este trecho de código é apresentado a seguir. Foram utilizados números para a indicação do estado para melhor tratamento pelo *software* criado, que será explicado na Seção 7.1.

```
// Inicialização e teste dos sensores
//Teste BMP180
Serial.println("Testando Sensores:");
Serial.print("BMP: ");
apc220.println("TestandoSensores:");
apc220.print("BMP: ");

if (BMP.begin())
 {
        // 1 = Sensor OK
        Serial.println("1");
        apc220.println("1");
}
else
```

```
{
        //2 = Sensor não encontrado. Checar Fiação
        Serial.println("2");
        apc220.println("2");
}

// Teste DHT
Serial.print("DHT: ");
apc220.print("DHT: ");
intchk = DHT.read22(DHT22_PIN);
switch (chk)
{
        case DHTLIB_OK:
        //1 = Sensor OK
        Serial.println("1");
        apc220.println("1");
        break;

        case DHTLIB_ERROR_CHECKSUM:
        //2 = Falha na checagem da biblioteca
        Serial.println("2");
        apc220.println("2");
        break;

        case DHTLIB_ERROR_TIMEOUT:
        //3 = Sensor não encontrado
        Serial.println("3");
        apc220.println("3");
        break;
}
```

A aquisição dos valores pelo microcontrolador ocorre de maneira diferente para cada sensor, possuindo níveis complexidades diferentes. Sucintamente, explicar-se-á como o microcontrolador faz a aquisição de cada valor.

Para o sensor BMP180, responsável pelas medições de Temperatura e Pressão, foi utilizado o processo padrão indicado pelo fabricante. Inicialmente é feito teste, caso atendido, inicia-se as medições. Para a aquisição das medições são criadas três variáveis, sendo duas delas os valores pretendidos e uma intermediaria de *status* do processo. É feito o processo indicado e durante o processo de aquisição das medidas meteorológicas, é realizado este teste de status no processo do sensor, através do protocolo I2C. A resolução do sensor escolhida foi a máxima, consequentemente o *delay* de retorno dos valores é o maior possível, e de acordo com o datasheet, é aproximadamente 12,5 ms, que para a nossa aplicação, este valor de espera é irrelevante.

Para o sensor DHT22, responsável pela medição da Umidade, é necessário o teste realizado acima para atualização valor enviado pelo sensor. Caso esta condição não for atendida, o resultado do teste indica falha e o sensor envia apenas o primeiro valor que é o aquisitado. O processo de aquisição dos dados é feito a partir do comando indicado, e é realizado de maneira direta, sem necessidades de linhas de código auxiliares.

Para o sensor HC-SR04, responsável pela medição de precipitação, é feita uma leitura direta da distancia do refletor em comparação ao sensor, caso este valor for o padrão definido na montagem e na calibração, indica que o nível no reservatório do pluviômetro é zero. Caso este valor seja menor que o padrão, o *software* indicará quantos milímetros lieanres o refletor se elevou. Esta distancia é a quantidade de água dentro do recipiente.

Todo o código utilizado no microcontrolador Arduino, devidamente comentado, está no Anexo B.

## 5.2. Ambiente de Desenvolvimento - *Visual Studio Community 2015*

O programa escolhido para a criação do *software* de monitoramento será escrito através do programa gratuito *Microsoft Visual Studio Community* 2015, utilizado para desenvolvimentoe criação de aplicativos desktop, para a plataforma de celular *Windows Phone* e aplicativos *WEB*. Dentro do *Visual Studio* existem alguns tipos diferente de linguagem de programação, alguns exemplos são o *Visual basic, Visual C#, Visual C++, JavaScript, Python*, entre outros [22].

A linguagem escolhida para o desenvolvimento deste projeto foi o *Visual Basic*, por apresentar códigos de maneira mais intuitiva e também por já o conhecimento inicial por parte dos desenvolvedores deste Projeto de Graduação.

### *5.2.1. Instalando o Software Visual Studio Community 2015*

Para instalar o Visual Studio no computador, faz-se necessário baixar o instalador diretamente do site oficial, e depois de terminado o *download*, é necessário a atualização de outro *software.NET Framework* para a sua versão mais atual, também disponível gratuitamente através da internet. Após o *download* destes dois *softwares*, seguimos com a instalação recomendada e concluímos o processo de instalação.

*5.2.2. Executando o software de interface de desenvolvimento*

Para a utilização do software não é necessária mais nenhuma instalação, sendo apenas requisitada a execução do programa no ícone criado "*Visual Studio 2015.exe*". Ao abri-lo, é exibida uma tela onde criamos um novo projeto, e dentro de uma nova tela exibida, é escolhida qual linguagem de programação vai ser desenvolvido o programa. Precisamos escolher a opção "*Windows Form*" para a criação de um aplicativo executável nos computadores *desktop* e *laptop*. A linguagem escolhida neste projeto foi a linguagem *Visual Basic* por prévio conhecimento dos alunos deste projeto de graduação. Nas Figuras 48 a 51, apresentamos esta escolha.

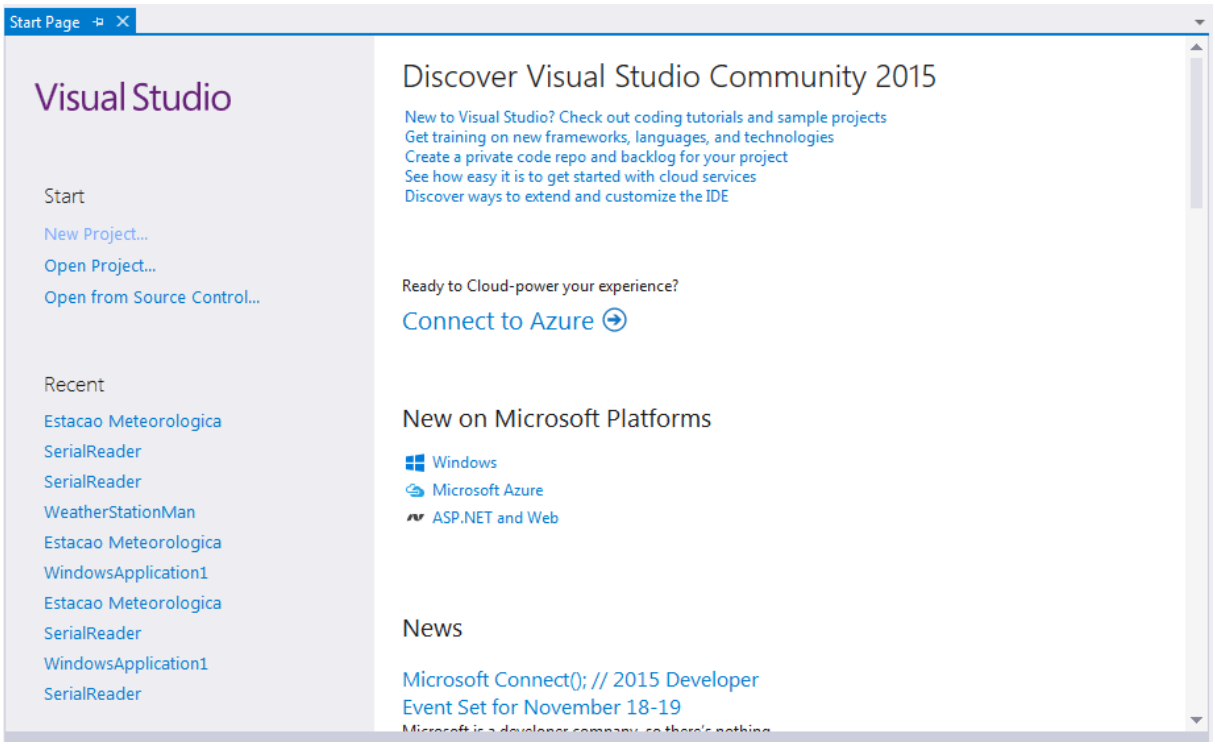



Figura 48 – Tela inicial do Visual Studio 2015.
Fonte: Autor.

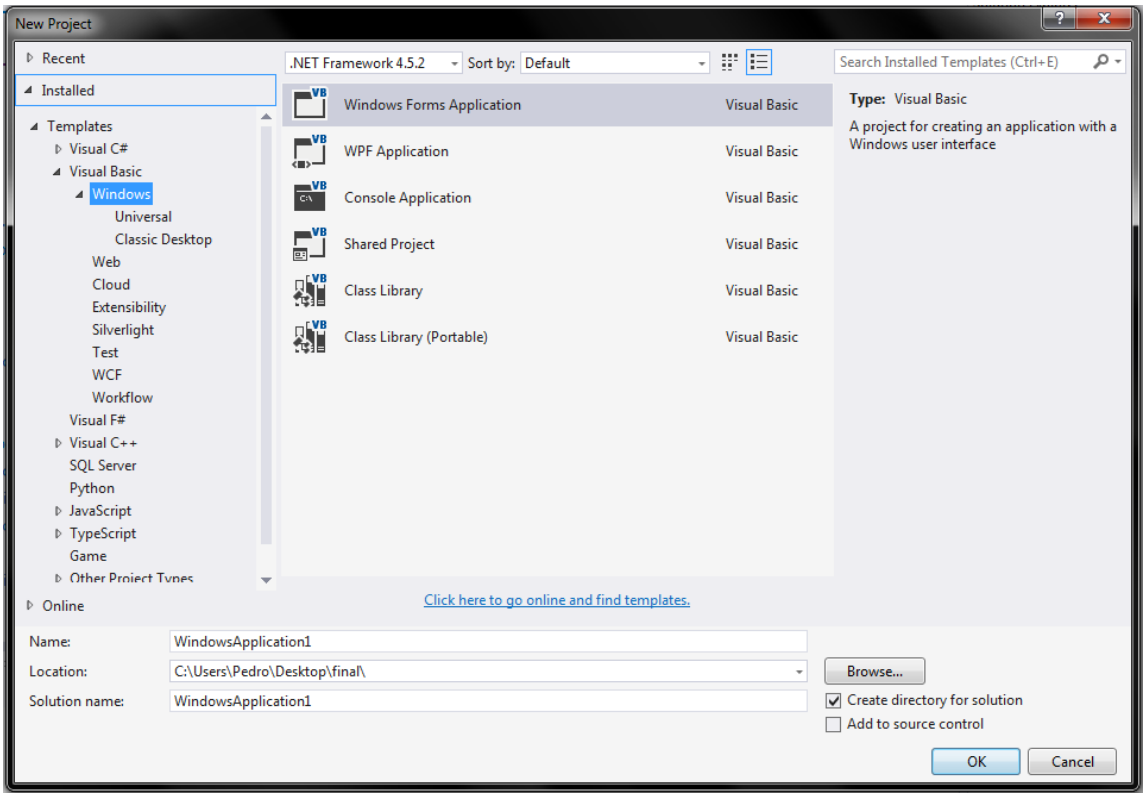



Figura 49 – Tela de escolha da linguagem.
Fonte: Autor.

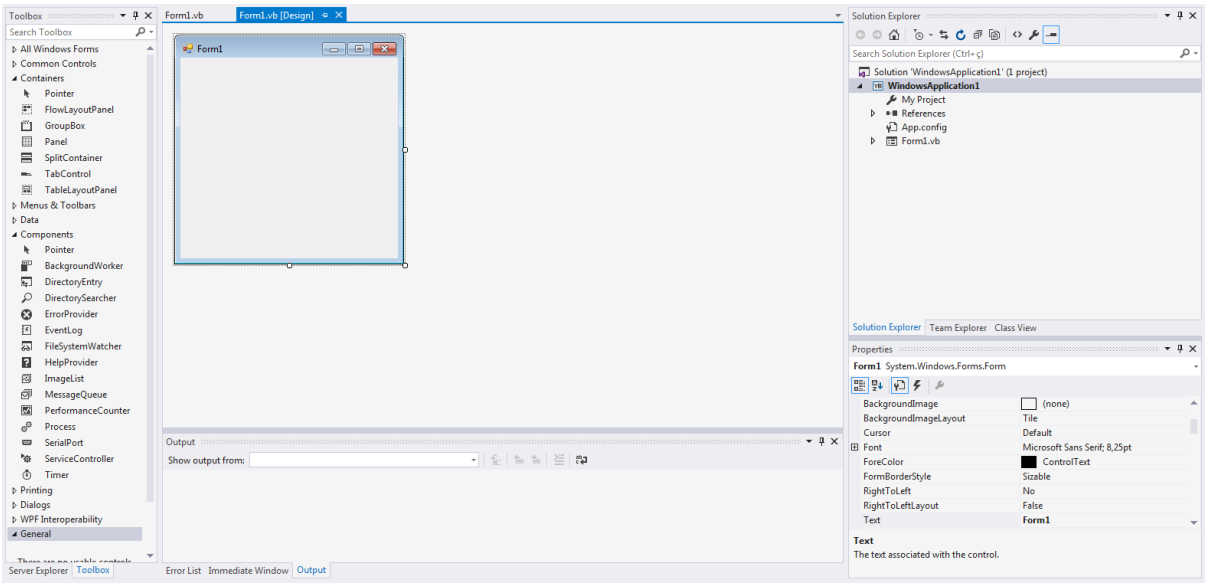



Figura 50 – Aparência inicial do aplicativo.
Fonte: Autor.

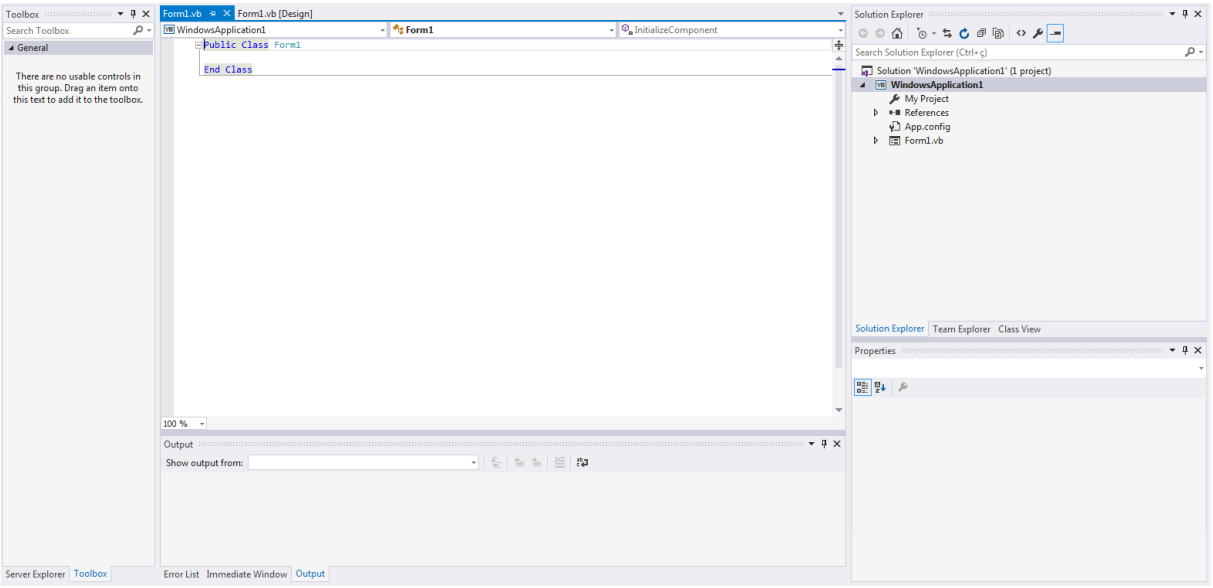



Figura 51 – Tela inicial de comandos do aplicativo.
Fonte: Autor.

### *5.2.3. Código Implementado – Visual Basic*

Todo o código utilizado está no Anexo C e será apresentado de forma mais explicativa na seção 7.1. Sucintamente, este código trata da recepção e exibição dos dados pelo módulo APC220, através da porta serial, no computador local. A plataforma *Visual Studio* é a responsável pela leitura dos dados na porta serial, que também precisa estar definida com o mesmo parâmetro do módulo de comunicação e do Arduino como sua velocidade de comunicação. Em seguida, estes valores são exibidos em uma área destinada para cada tipo de medida, e também uma planilha onde são registrados com sua marca de tempo, feita via *software*. É possível também gerar gráficos, para avaliação visual da alternância dos dados. Os dados podem ser salvos em um arquivo com extensão ".csv"(*Comma-Separated Values*), onde podem ser manipulados por diversos outros *softwares*, caso seja necessário [23].

## 5.3. Interface Gráfica

Conforme o desenvolvimento do projeto no *Visual Studio*, criou-se a interface gráfica, onde seu objetivo é fornecer uma resposta visual ao usuário do programa que esteja rodando, onde toda a navegação seja facilitada através de botões, janelas, gráficos, todos estes respondendo aos comandos do operador.

Neste projeto a interface gráfica foi criada com a intenção de monitorar as variáveis definidas anteriormente neste projeto, registrar com estampa de tempo quando foram aquisitadas via *software*, geração de gráficos e armazenamento dos dados automaticamente e a partir de requisição do usuário, todos estes comandos de forma bem intuitiva. Todo o funcionamento fica disponível através da escolha da porta serial e da escolha da velocidade de conexão que está configurado o módulo APC220. A Figura 52 exibe a interface criada para o monitoramento.

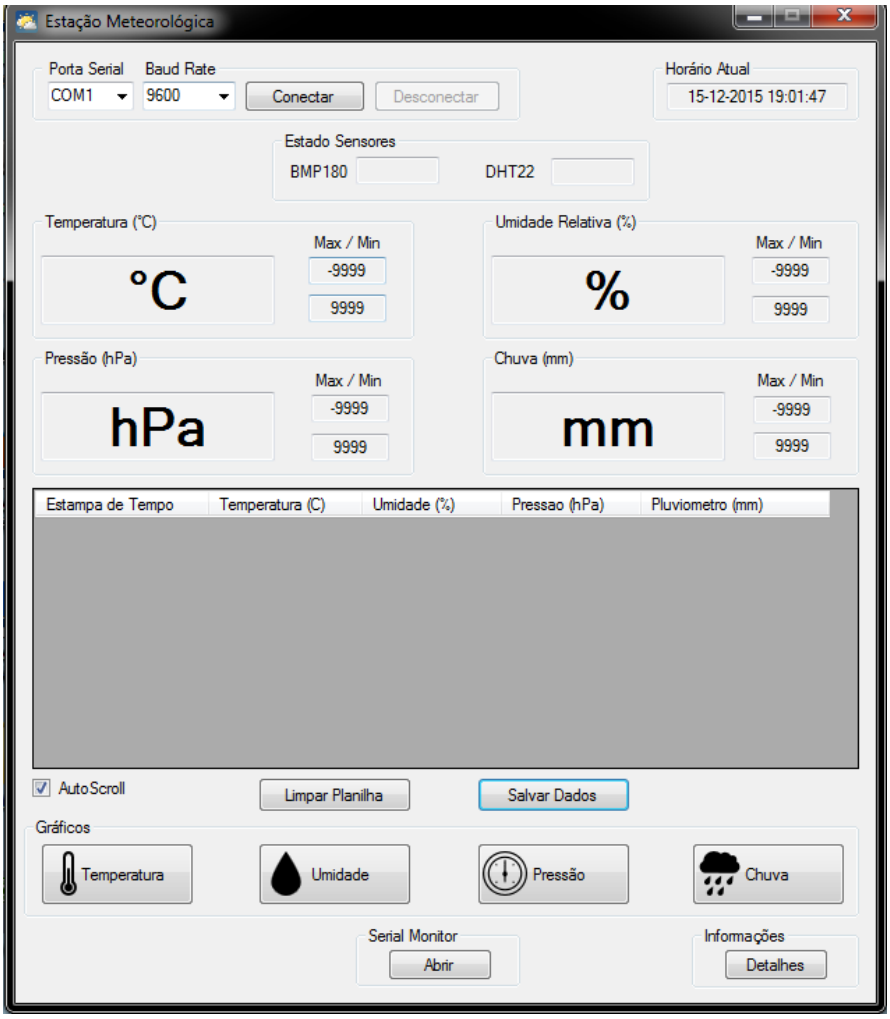



Figura 52 – Interface criada para o monitoramento.
Fonte: Autor.

Como é possível ver na Figura 52, existem quatro *textBox* (componente disponibilizado pelo *software* para indicação de caracteres) que exibem os quatro valores atuais coletados pelo Arduino, além de mais oito *textBox*, dois para cada sensor, que exibem o valor mínimo e máximo coletado durante a execução do *software*. Também foi criada uma planilha que exibirá os valores

com estampa de tempo do momento da recepção via *software*, dois botões que, respectivamente, limpam a planilha e salva os dados, além de botões que criam gráficos a partir destes valores da planilha.

Além das informações dos sensores, existe a barra de conexão com a porta serial, onde são exibidas todas as portas COM disponíveis, e para o funcionamento selecionamos a porta em que o receptor foi instalado e o *baud rate*, onde este valor atende aos possíveis valores pré-determinados pelo módulo RF de comunicação.

Todo o funcionamento do *software* criado será detalhado na subseção 7.1.

# 6. TESTES E CALIBRAÇÃO DOS SENSORES

Neste capítulo trataremos testes, calibração e integração dos módulos do projeto, partindo do circuito eletrônico do *hardware* definido e do *software* finalizado, restando como escopo do desenvolvimento, verificação do funcionamento dos sensores, consistência dos valores medidos, funcionamento do módulo de comunicação e apresentação destes dados pelo *software*.

## 6.1. Sensor de Temperatura e Pressão BMP180

Este sensor, que possui mais de uma unidade, vem com calibração de fábrica. Em seu manual é explicado como essa calibração é feita através da leitura de bits feita pelo sensor. Então foi desenvolvido um algoritmo que segue o procedimento e recomendação do fabricante para a sua calibração. Como este algoritmo de calibração é feito externamente pelo Arduino, então não alteramos a calibração de fábrica do sensor caso esta esteja errada. Se feita corretamente, esta calibração criada será usada para conferir com os valores aquisitados pela configuração de fábrica do sensor. O processo de calibração feita pelo fabricante é exibido na Figura 53. Todo o código do Arduino está no Anexo B.

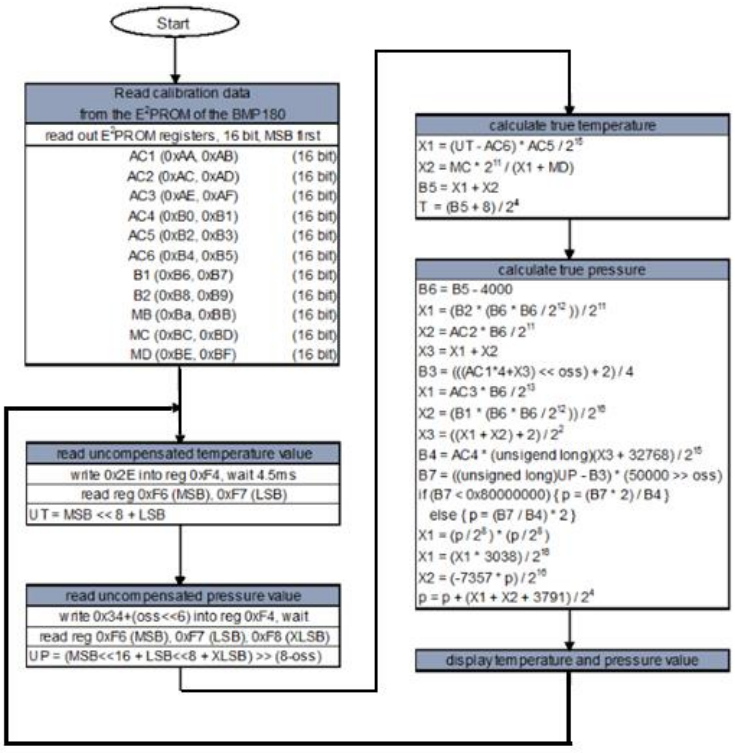



Figura 53 – Algoritmo de calibração do fabricante do sensor de temperatura e pressão BMP180. Extraído de [17].

Ao baixar a biblioteca para o funcionamento do sensor, exemplos de teste padrão do fornecedor também são instalados no *software* do Arduino, criando assim, um parâmetro para conferirmos se o código desenvolvido está de acordo com padrão de fábrica. Nas Figuras 54 e 55 a seguir, veremos os exemplos padrão fornecidos para os valores de Altitude em relação ao nível do mar, Temperatura e Pressão.

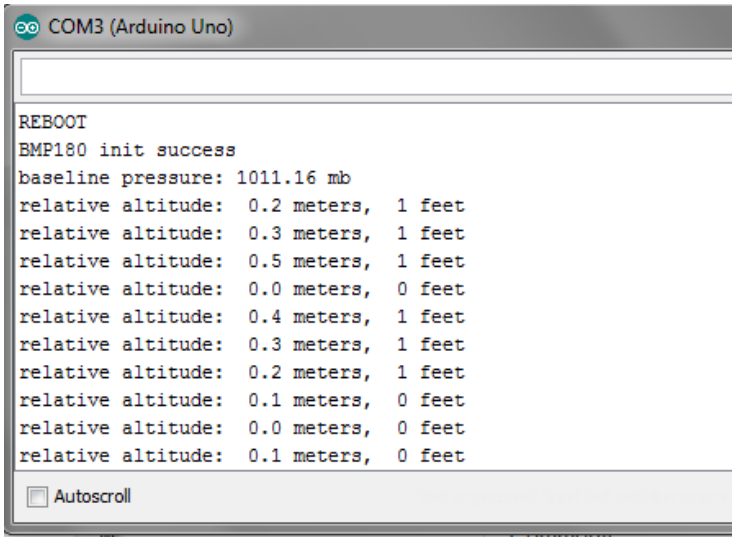



Figura 54 – Valor de Altitude Relativa dada pelo exemplo padrão.
Fonte: Autor.

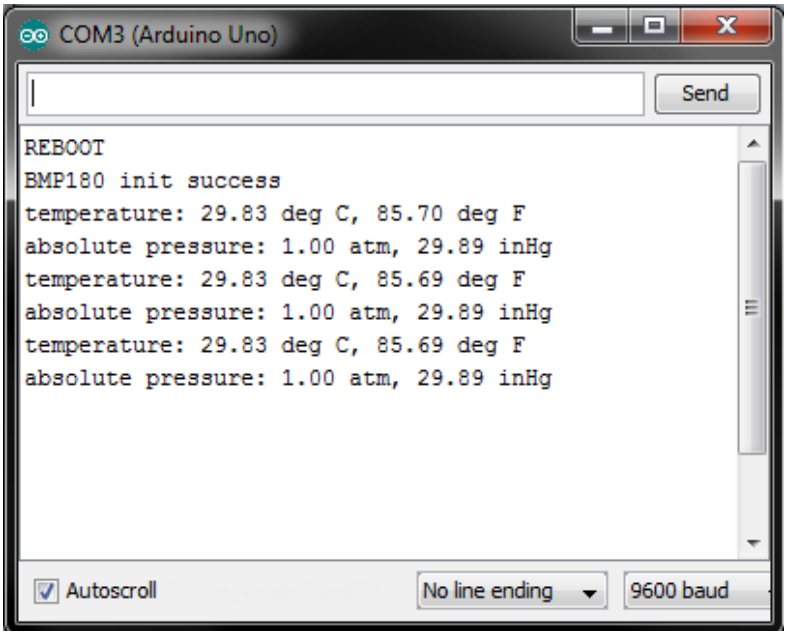



Figura 55 – Valores de Temperatura e Pressão dado pelo exemplo.
Fonte: Autor.

Considerando estes valores como base, podemos executar o código de calibração que aquisita os dados deste sensor e, para saber de sua consistência, é feita comparação com os valores base. Na Figura 56, é exibida a leitura de valores feita pelo sensor através do algoritmo de calibração criado.

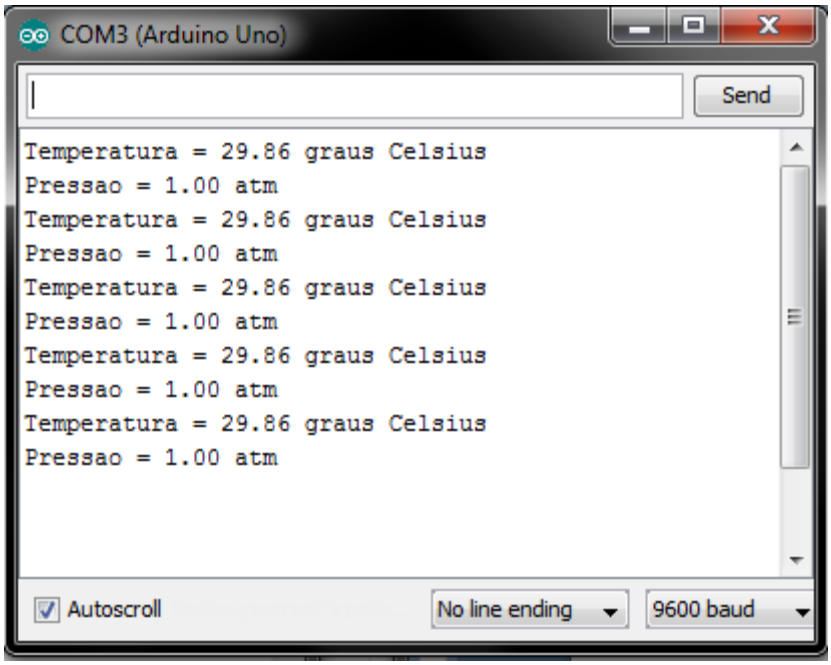



Figura 56 – Valores fornecidos pela leitura do Algoritmo de calibração criado.
Fonte: Autor.

Podemos observar que existe uma diferença na segunda casa decimal na medida de temperatura, que pode ser tratada como diferença real de temperatura quando os códigos foram executados, ou como erro, quando a resolução deste sensor é ± 0,5 °C, informado pelo fabricante, tornando assim, esta diferença praticamente irrelevante para a aplicação deste projeto.

Concluímos que o código criado no *software* funcionou corretamente e a calibração externa do sensor BMP180 considerada um sucesso.

## 6.2. Sensor de Umidade DHT22

Este sensor, que possui mais de uma unidade, porém só está sendo utilizada a captação de Umidade, vem com calibração de fábrica. Em seu manual é explicado que este sensor como é feita a calibração. Em poucas palavras, este processo de calibração fabril é feita em uma câmara com

umidade extremamente especifica, armazenando assim, estes coeficientes de calibração no sensor que irão responder aos pedidos do microcontrolador de leitura de dados.

Um calibrador externo que poderia ser utilizado para o tratamento deste sensor seria através de um psicrômetro, que é um aparelho que contém dois termômetros fixados sobre um suporte único. Um termômetro possui o bulbo seco e o segundo, bulbo molhado. O aparelho é apresentado na Figura 57.

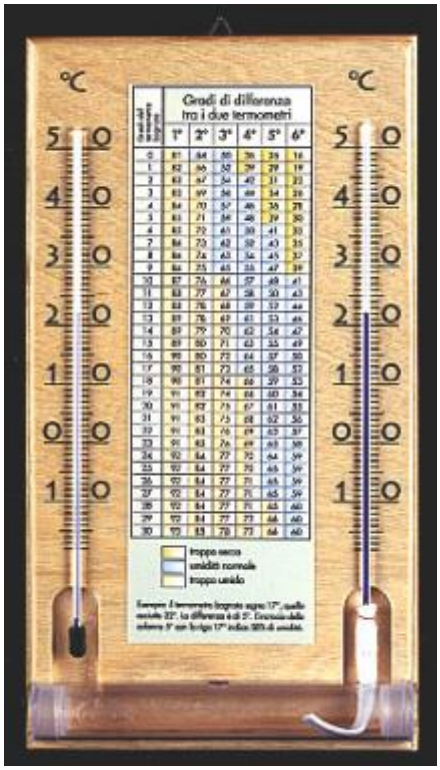



Figura 57 – Psicômetro.
Extraído de [24].

Por se tratar de um aparelho incomum, encontraram-se dificuldades na disponibilidade deste calibrador com intuito de emprestimo e/ou utilização do mesmo para atender a necessidade abordada neste tópico.

O processo de medida do psicometro consiste na comparação das temperaturas entre os termometros de bulbo seco e bulmo molhado, onde este apresenta temperatura um pouco menor devido a sua composição. Esta diferença, dada com propriedades da mistura do ar e do vapor d'agua são apresentadas de forma gráfica, no que denomina-se carta psicrométrica [24].

## 6.3. Sensor de Distância HC-SR04

Após pesquisas, descobrimos que não existe calibração propriamente dita para sensores ultrassônicos. Como dito anteriomente, o sensor usa o tempo entre a emissão da onda e a recepção do echo para calcular a distancia. Este sensor usa a velocidade do som para o cálculo e ao contrario dos outros sensores, esta velocidade do som não varia em área aberta com a variação das condições naturais em que ele se encontra.

Caso fosse encontrado algum problema no sensor, seria causado ou por má ligação ou por um sensor danificado. A seguir, faremos alguns testes com o sensor tentando levantar uma curva de precisão e após esse teste, o usaremos no recipiente do pluviômetro.

O procedimento de teste de precisão foram feitas três leituras de distância do refletor pelo sensor de acordo, e o resultado esperado é um reta de inclinação 1, apresentada na Figura 58. A operação do sensor inicia-se a partir de 2 centimetros porque é o valor mínimo de funcionamento estipulado pelo fabricante e para evitar leituras erradas, a priemira medição ocorrerá em 2,5 centimetros. A montagem e os resultados são mostrados nas Figuras 58 e 59 e na tabela 9.

*Tabela 9 – Medidas feitas nos testes de calibração do sensor HC-SR04.*

| Valor esperado (mm) | Valor medido (mm) |
|---|---|
| 2,50 | 2,51 |
| 2,50 | 2,50 |
| 2,50 | 2,51 |
| 3,00 | 3,00 |
| 3,00 | 3,01 |
| 3,00 | 3,00 |
| 4,00 | 3,98 |
| 4,00 | 3,99 |
| 4,00 | 3,99 |
| 5,00 | 5,01 |
| 5,00 | 5,01 |
| 5,00 | 5,00 |
| 6,00 | 5,98 |
| 6,00 | 5,99 |
| 6,00 | 6,00 |
| 7,00 | 7,00 |
| 7,00 | 7,00 |
| 7,00 | 7,00 |
| 8,00 | 8,01 |
| 8,00 | 7,99 |
| 8,00 | 8,00 |

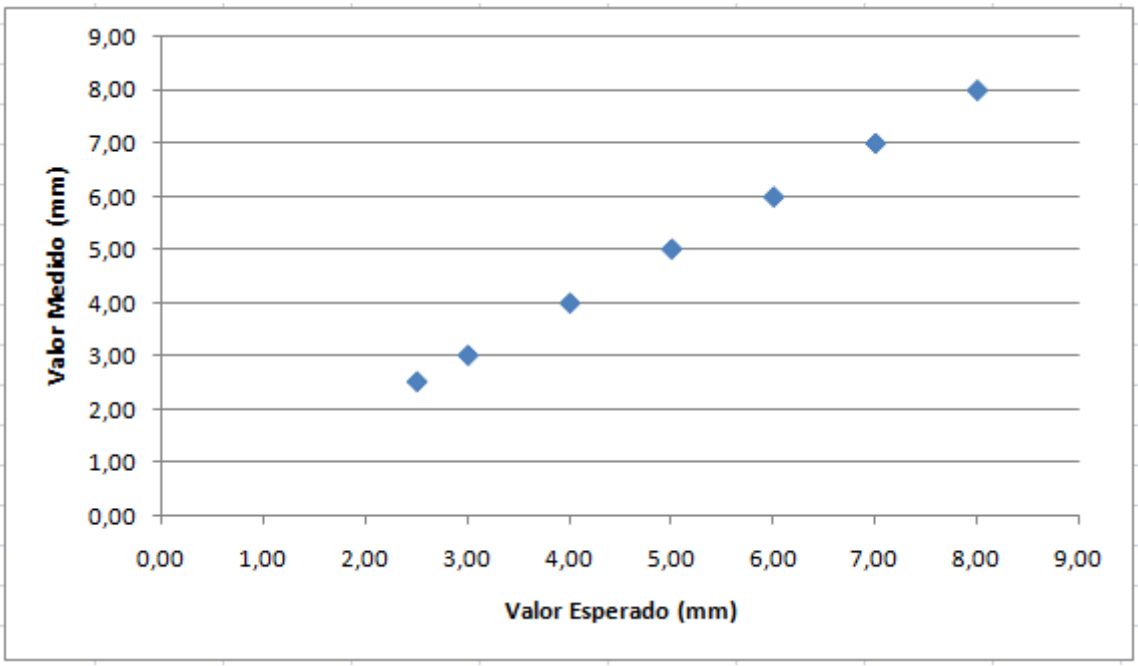
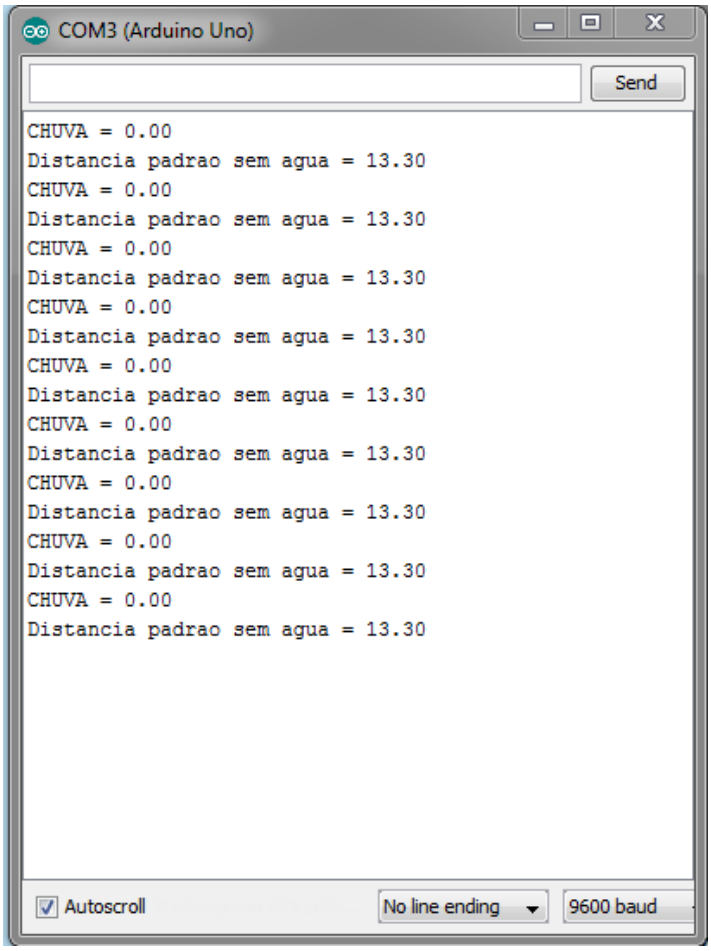



Figura 58 – Gráfico dos Valores Esperados x Valores Medidos.
Fonte: Autor.

Figura 59 – Método de realização do teste.
Fonte: Autor.

Na Figura 60, mostraremos a distancia padrão do sensor ao refletor sem água e usaremos essa distancia na calibração para indicar nível zero no pluviômetro.




(a) (b)

Figura 60 (a) e (b) – (a) Recipiente vazio; (b) Resultado da calibração da altura padrão com o recipiente vazio.
Fonte: Autor.

Nas Figuras 61 e 62 mostraremos o pluviômetro com certo nível de água e a leitura do sensor deste nível.

(a) (b)

Figura 61 (a) e (b) – Recipiente exibindo 21 mm de altura.
Fonte: Autor.

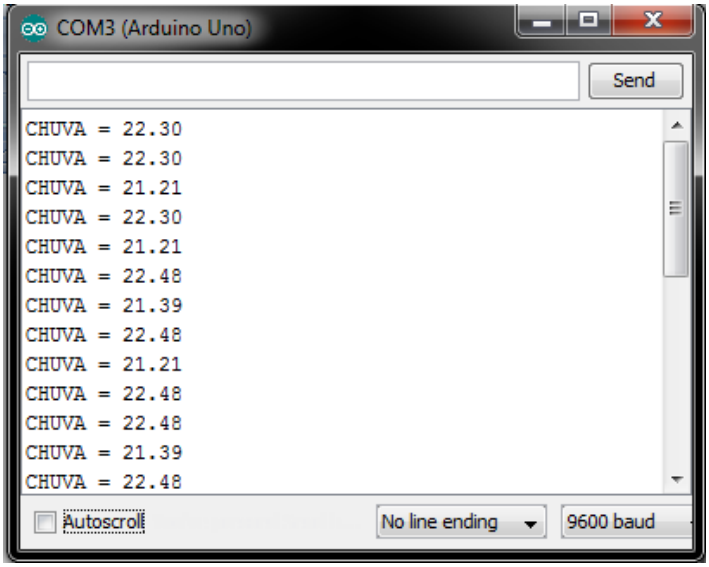



Figura 62 – Teste do pluviômetro com o recipiente com água.
Fonte: Autor.

Como no Arduino não tem uma função específica para arredondamento, e por padrão do sensor, o valor é enviado com duas casas decimais, verficamos uma vriação costante no valor da parte fracionária do sensor, que a principio é o decimal de milímetro, praticamente imperceptível a olho nu. A seguir, no *software* de monitoramento criado, existe a possibilidade de um comando de arredondamento, a medida será exibida sem essa parcela decimal.

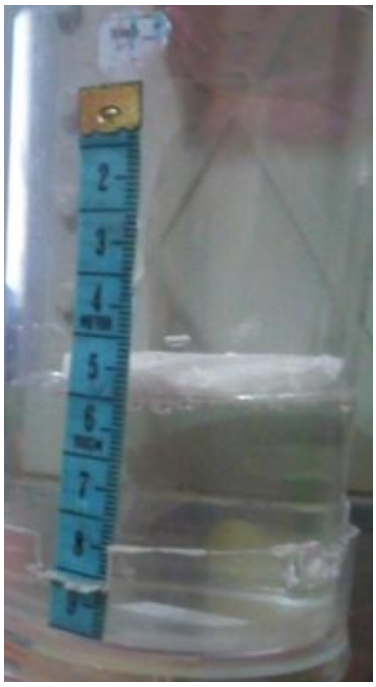


Figura 63 – Recipiente exibindo aproximadamente 40 mm de altura.
Fonte: Autor.

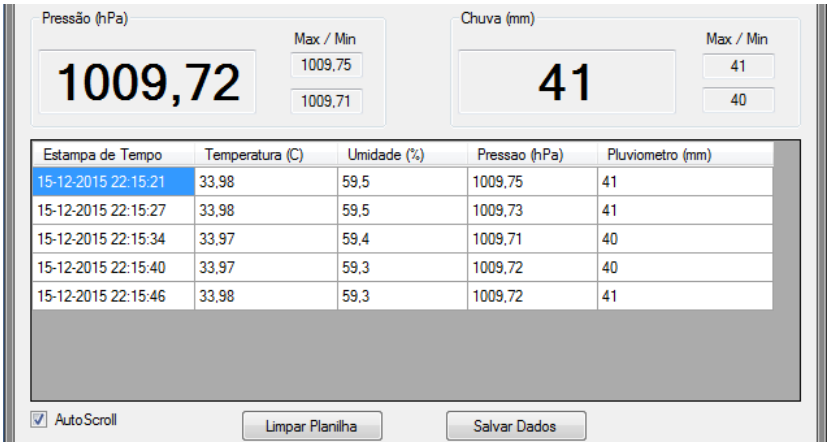



Figura 64 – *Software* exibindo a informação do pluviômetro.
Fonte: Autor.

## 6.4. Configuração e Teste do Módulo de Comunicação APC220

O módulo RF Wireless APC220 contém um *software* especifico para sua configuração, o RF-Magic. Seus parâmetros de funcionamento foram estudados e apresentados no Capítulo 3. As configurações para seu funcionamento são exibidas na Figura 63 abaixo.

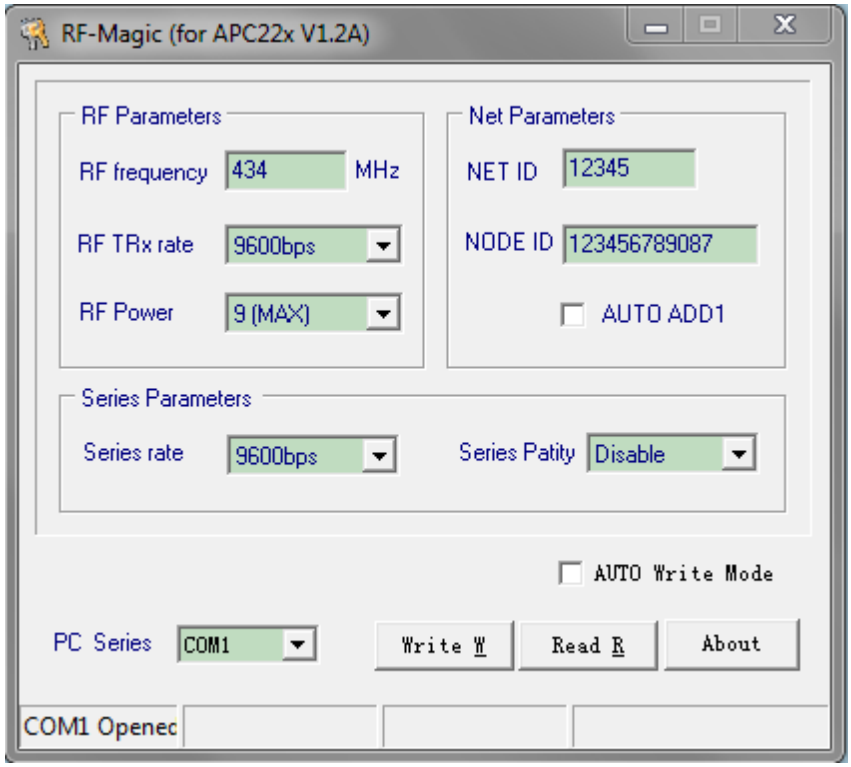



Figura 65 – *Software RF-Magic*.
Fonte: Autor.

Conforme apresentado no Capítulo 3, na Figura 25, o APC220 não possui uma maneira direta de conexão com o computador. Então, fez-se necessário autilização de um conversor TTL-USB, e foi usado o conversor USB-TTL CP2102, que é a responsável pela interligação entre o módulo e o computador. Apresentaremos na Figura 66 o conversor e na Figura 67 o conversor com o módulo no computador [25].

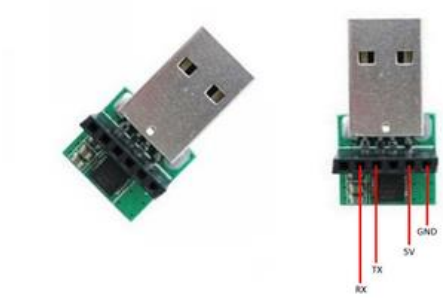



Figura 66 – Conversor *USB-TTL CP2102*.
Extraído de [25]

Figura 67 – Conversor USB-TTL CP2102 em conjunto com o módulo de comunicação APC220.
Fonte: Autor

Os parâmetros de frequência e de *baud rate* foram definidos após uma breve pesquisa no datasheet para o melhor funcionamento do módulo, então foram definidos os valores de 434 MHz para a frequência, valor que se propaga ondas eletromagnéticas no Brasil sem grandes interferências e 9600 bps de *baud rate*, que é a velocidade de bits por segundo em que o módulo atinge sua distância máxima de operação.

Os *Net parameters* são para codificação da transmissão, para que não haja conflito na hora do envio e recepção de dados, onde *Net ID* é o valor que define um código de rede e *Node ID* o valor que define um código de cada módulo de comunicação APC220.

Os *Series parameters* são parametros de recuperação de dados, onde os valores de *Series Rate* e *Series Parity* são os dois parametros que controlam esta ação. Muito utilzados em

arquiteturas RAID (*Redundant Array of Independent Disks* - Conjunto Redundante de Discos Independentes), é uma forma de criar um subsistema de amazenamento composto por vários discos. Na nossa aplicação foram deixados os valores padrões [26].

O teste da comunicação foi realizado de acordo com um código criado no arduino, em que é enviada uma mensagem de texto lida pela porta COM dizendo que o módulo está em funcionamento. O código do Arduino deste programa está no Anexo A. Nas figuras 68 e 69 a seguir é apresentado, com o auxílio de um software chamado *Serial Port Utiliy* [27], os dados sendo enviado pela porta serial (COM3) e pelo módulo de comunicação APC220 (COM4).

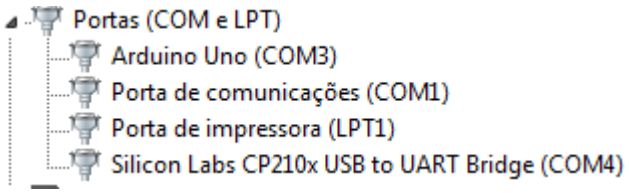



Figura 68 – Identidicação das portas COM do Arduino e do Módulo APC220.
Fonte: Autor.

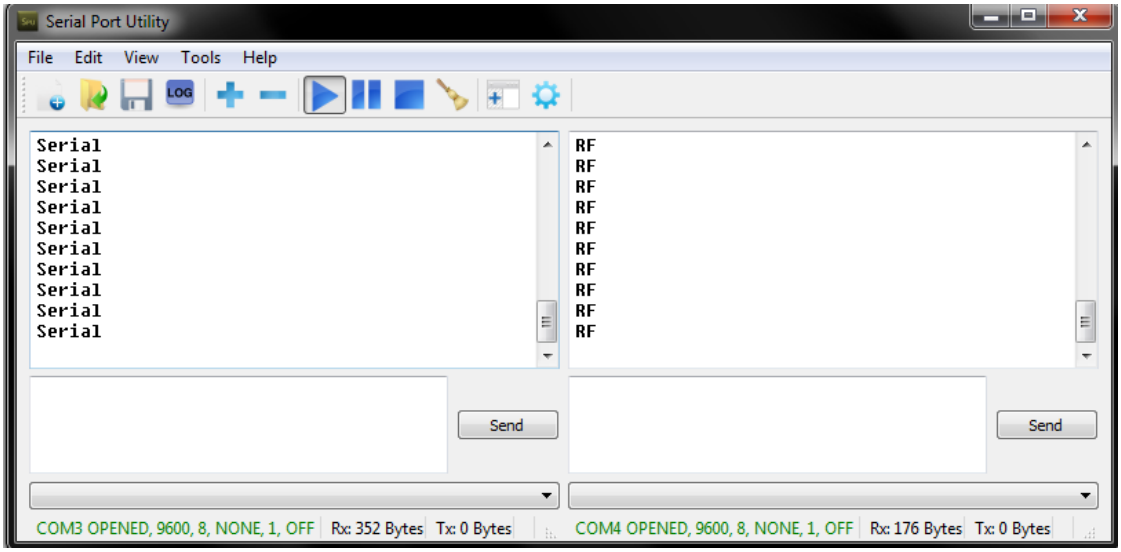



Figura 69 – *Software Serial Port* exibindo as mensagens.
Fonte: Autor.

## 7. TESTES DO PROTÓTIPO – RESULTADOS, ANÁLISE E DISCUSSÃO DE DADOS

No capítulo 6, concluímos que todos os módulos pertencentes à estação meteorológica estão calibrados e em funcionamento. Assim, todos os pré-requisitos foram atendidos para o funcionamento do *software* de monitoramento. A Figura 70 apresenta o diagrama de blocos de funcionamento de todo o projeto da Estação Meteorológica.

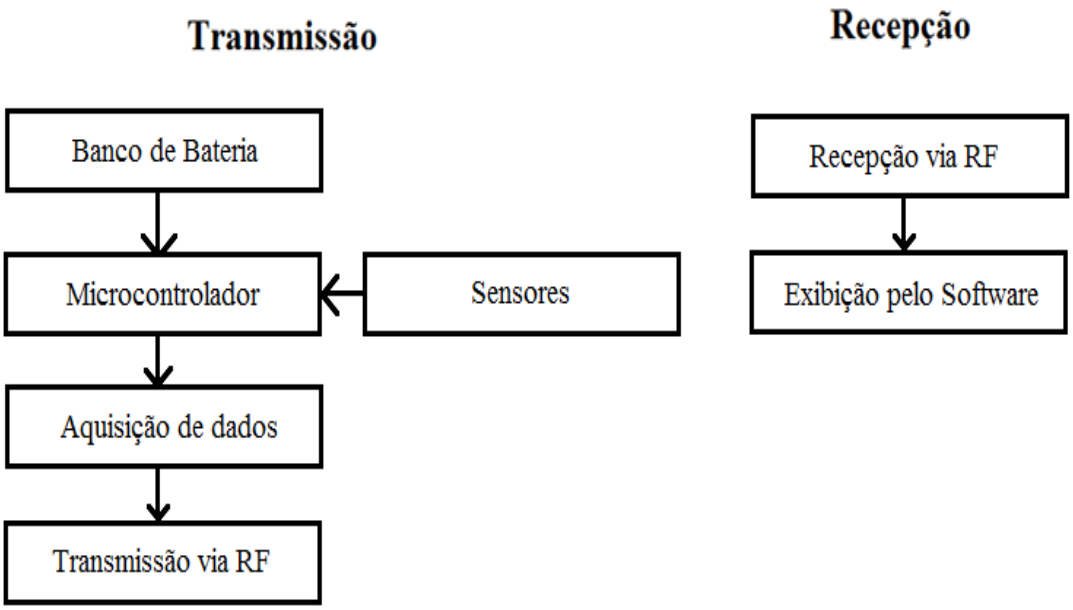



Figura 70 – Diagrama de blocos de funcionamento da Estação Meteorológica.
Fonte: Autor.

O diagrama explica sucintamente que ao aplicarmos a alimentação externa ao microcontrolador e, indiretamente, aos sensores que estão sendo alimentados pelo microcontrolador, é feito um teste de rotina para a o início da operação destes sensores. Caso algum sensor responda de forma inválida, é necessária a verificação da alimentação e/ou interligação com o Arduino deste sensor. Caso estejam todos em funcionamento, inicia-se a operação de coleta de dados e seguidamente é feita a transmissão via *wireless* pelo módulo de comunicação. Com o par de módulo de comunicação APC220 devidamente configurado, a unidade que está conectada ao computador começa a receber os dados. Então o *software* de monitoramento é executado e exibe todas as variáveis da estação.

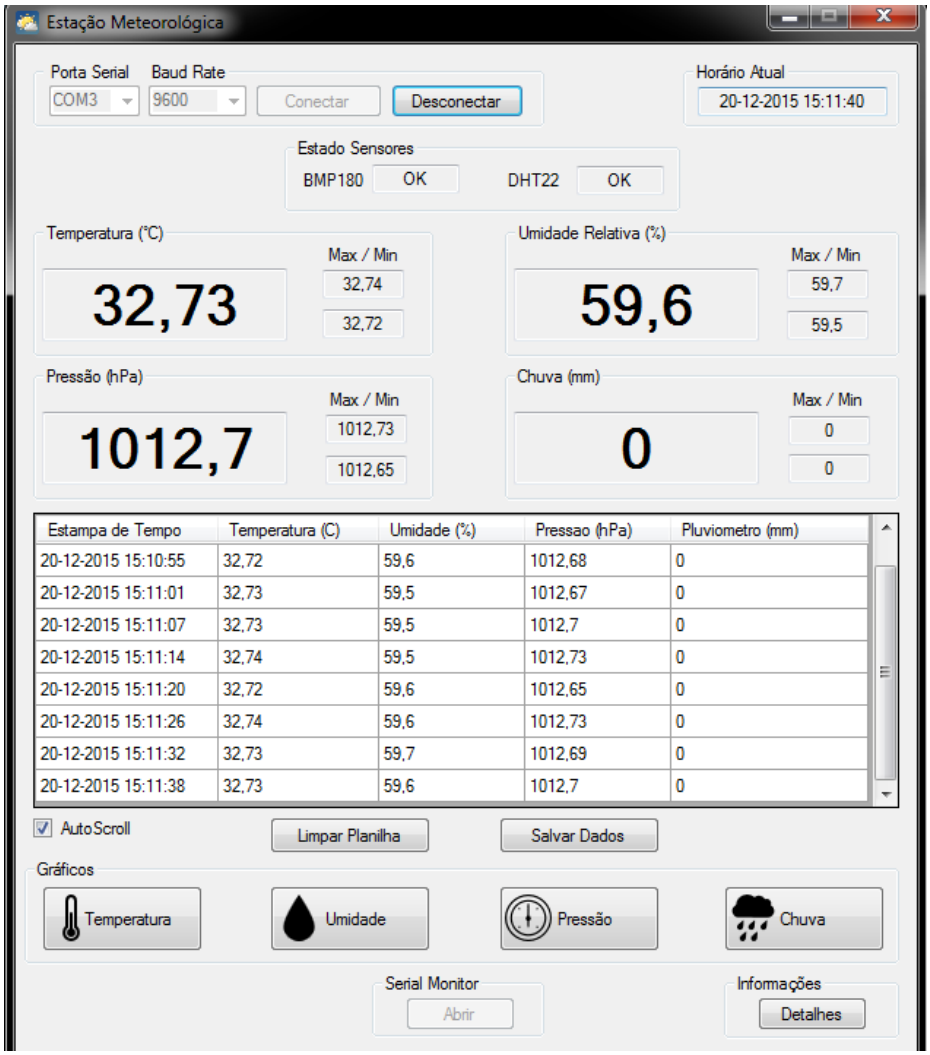



Figura 71 – *Software* de Monitoramento.
Fonte: Autor

## 7.1. Integração do *Hardware* com o *Software*

Abordaremos nesta seção como o microcontrolador transmite e a recepção pelo módulo juntamente com o *software*.

O Arduino está executando o código final do projeto, onde aquisita os dados e os exibe de maneira simplificada, sendo cada variável medida identificada por uma abreviação do nome da medida, o que denominaremos de etiqueta, antes da exibição. Estes dados são enviados em linhas diferentes para facilitar a recepção e tratamento dos mesmos, que será explicada mais a frente nesta seção. A Figura 72 exibe como o Arduino está enviando os dados pela porta serial e pelo módulo de comunicação.

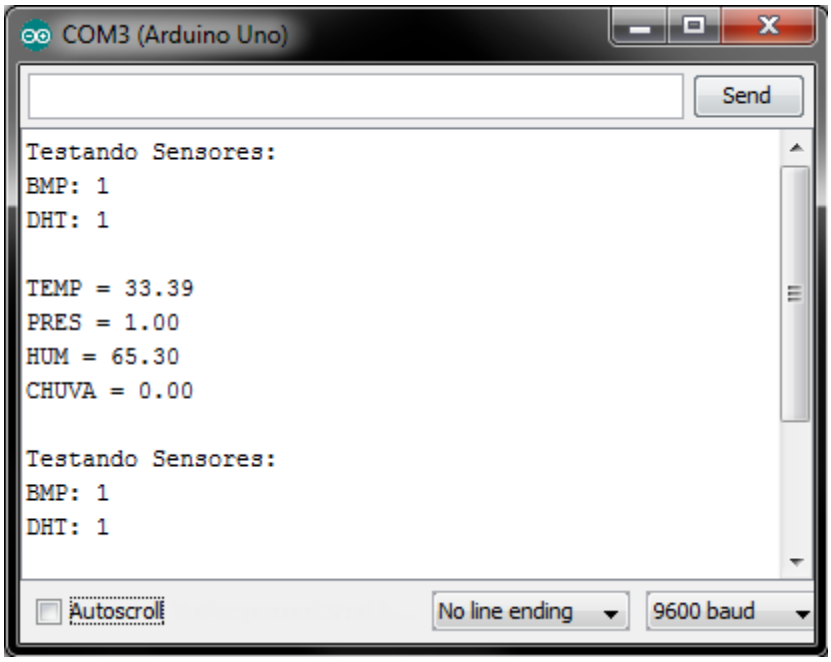



Figura 72 – *Serial monitor* do Arduino.
Fonte:Autor.

Este código no Arduino foi desenvolvido para enviar os dados pela porta serial e pelo transmissor APC220. Como vimos no Capítulo 3, o Arduino a princípio, possui somente um par TX/RX, mas com a inclusão da biblioteca *SoftwareSerial.h*, podemos tornar dois outros pinos digitais quaisquer em outro par TX/RX, e assim o fizemos para termos condições de utilizar o APC220 de forma constante sem a necessidade de retirar sua fiação quando precisarmos alterar o código Arduino [28].

O programa criado apresenta também seu próprio *Serial Monitor*, que é uma tela de monitoramento dos dados que atravessam a porta serial, para que o operador necessite utilizar apenas um *software* para todo o monitoramento. Este recurso foi acrescentado no *software* para que, caso o programa não esteja apresentando às informações adequadamente, existe a opção de monitorar o que está sendo recebido pela porta serial através do módulo de comunicação, sem a utilização de um *software* secundário, seja ele do Arduino ou qualquer outro. A Figura 73 apresenta a tela criada.


Figura 73 – *Serial monitor* criada.
Fonte:Autor.

Para a recepção dos dados pelo *software*, é necessário a abertura da porta serial através do comando criado "Conectar()". Este comando abre a porta COM que foi selecionada pelo operador do programa, e permite que os dados transmitidos sejam lidos pelo *software*. Esta recepção também depende do *baud rate* configurado no código do Arduino e no módulo de comunicação APC220. Estes valores são pré-determinados pelo fabricante do módulo e caso não sejam iguais em todos os locais que são configurados, os dados se tornarão *garbage data* e são impossíveis de se traduzir. A Figura 74 mostra os parâmetros para a abertura da porta serial.

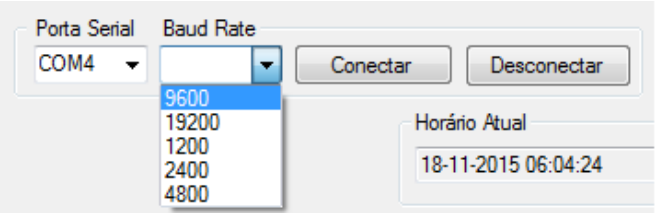



Figura 74 – Parâmetros para a conexão com a porta serial.
Fonte:Autor.

O fragmento do código que trata da conexão e desconexão da porta serial é apresentado a seguir:

```
Public Sub Desconectar()
	If comOpen = True Then
		SerialPort1.Close()
	End If
End Sub

Public Sub Conectar()

'Propriedades da porta serial tem que estar de acordo com o configurado no par
transmissor/receptor
	With SerialPort1
		.PortName = cbPortas.Text
		.BaudRate = cbBaud.Text
		.Parity = IO.Ports.Parity.None
		.DataBits = 8
		.RtsEnable = False
		.ReadTimeout = 300000	'Tempo maximo sem dados = 5 min
	EndWith

	'Concetando-se a porta serial
	Try
		SerialPort1.Open()
		comOpen = SerialPort1.IsOpen
	Catch ex AsException
		comOpen = False
	End Try
End Sub
```

Para que a porta serial não fique aberta quando encerramos a execução do programa, foi incluído este mesmo trecho de código “Desconectar()” quando encerramos o programa sem clicarmos no botão Desconectar, tornando assim o programa seguro em relação a questão de abertura e fechamento da porta serial.

A leitura dos dados realizada pelo *software* é feita pelo trecho de comando a seguir. Como visto no código implementado no Arduino, para todo valor existe uma etiqueta, que são TEMP,

HUM, PRES ou CHUVA. O software lê os dados, analisa a sua etiqueta e chama uma subrotina de atualização do *textBox* da medidacom o valor correspondente da variável. Este indicador é removido através da linha de código *dados = dados.Replace("TEMP = ", "")*, onde esta subrotina de atualização dos dados, que é apresentada abaixo, executa o seguinte procedimento:

```
If comOpen Then
            Dim dados As String = SerialPort1.ReadLine()

            If dados.Contains("TEMP") Then
                dados = dados.Replace("TEMP = ", "")
                If IsNumeric(dados) Then
                    atuaTemp = CDbl(dados / 100)
                    tbTemp.Invoke(New UpdateTempBoxDelgate(AddressOf UpdateTempBox),
atuaTemp.ToString)
                End If

Delegate Sub UpdateTempBoxDelgate(ByVal t As String)

    'Atualiza Valor de Temperuta
    Public Sub UpdateTempBox(ByVal t As String)
        tbTemp.Text = t

        'Temperatura máxima
        If tempMax.Text < tbTemp.Text Then
            tempMax.Text = tbTemp.Text
        End If

        'Temperatura mínima
        If tempMin.Text > tbTemp.Text Then
            tempMin.Text = tbTemp.Text
        End If
    End Sub
```

É criada uma variável temporária de dados, que é a leitura da linha enviada pela porta serial. Assim que os dados são lidos, ocorre uma identificação da variável pela sua etiqueta, no caso deste exemplo usamos a atualização da temperatura, então esta etiqueta é removida. Estes dados vão para outra variável intemediaria ocorre uma conversão de *float point*, forma do dado vindo do Arduino, para *double data* para exibição no *textBox*, onde nesta conversão é perdido a casa decimal, por isso os dados são divididos por 100, e após esta conversão é chamada a subrotina de atualização do *textBox* correspondente.

A escrita dos dados na planilha, que no código é chamada de *DataGridView1*, é feita assim que os quatro valores das medidas são exibidas nos *textBox* correspondentes. Esta planilha conta com cinco colunas, uma para cada variável medida e a última para o momento em que os dados foram recebidos pelo *software*. Na atualização do *textBox* de chuva, adiciona-se uma linha nesta planilha com todos os valores. Esta adição é feita com o recebimento do valor do índice pluviométrico, porque no envio dos dados feito pelo Arduino, este valor é o último a ser enviado,

garantindo assim que quando este dado é recebido, todas as outras variáveis medidas também foram. O trecho do código que realiza este tratamento de dados é apresentado a seguir:

```
Dim linha AsInteger = DataGridView1.Rows.Add
DataGridView1.Rows.Item(linha).Cells(0).Value = tbHora.Text
DataGridView1.Rows.Item(linha).Cells(1).Value = tbTemp.Text
DataGridView1.Rows.Item(linha).Cells(2).Value = tbHum.Text
DataGridView1.Rows.Item(linha).Cells(3).Value = tbPres.Text
DataGridView1.Rows.Item(linha).Cells(4).Value = tbChuva.Text
salvaCSV(DataGridView1, "Registro.csv")
```

Percebemos na última linha do trecho apresentado que existe um comando denominado *salvaCSV(DataGridView1, "Registro.csv").* Esta linha realiza um registro automático dos dados que estão na planilha toda vez que uma nova linha é adicionada, não precisando assim o operador do software executar qualquer tipo de atuação para o registro dos dados.

Como apresentado na Figura 50, a estação meteorológica possui dois botões de manipulação da planilha. Nesta manipulação, foi incluído um botão que salva os dados por comando do operador no *software*. Em uma primeira análise parece redundante, já que o software já salva automaticamente os dados. Mas este processo automático "Registro.csv" salva apenas os dados estão contidos no momento que estão no *DataGridView1*. Ao clicarmos no botão "Salvar Dados" é criado um novo registro onde o nome do arquivo contém data e hora que foi salvo, tornando assim, este registro de arquivo mais especifico. A Figura 75 a seguir é exibida quando clicamos no botão "Salvar Dados".

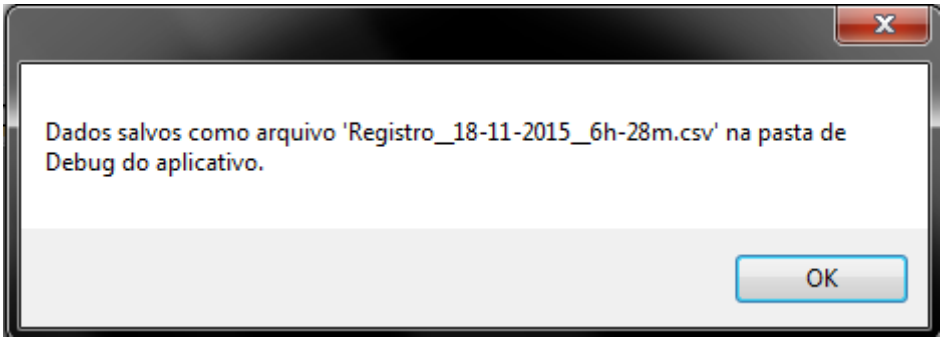



Figura 75 – Janela que exibe que o Registro manual foi feito.
Fonte: Autor.

Para maior segurança e confiabilidade, foi incluido um processo de confirmação de registro dos dados com especificação de data e hora quando fechamos o programa e quando clicamos no botão limpar planilha, tornando assim, o *software* protegido contra quaisquer descuidos do operador no que se trata sobre registro de dados, possuindo registro temporário e específico.

O processo de exibição dos gráficos é dado a partir dos dados coletados pela planilha, realizando a coleta de dados das variáveis desejada no eixo das ordenadas e utilizando a estampa de tempo no eixo das abscissas, este processo é feito a partir do trecho de código mostrado a seguir:

```
Public Class humiGraf
    Private Sub humiGraf_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        For Count As Integer = 0 To Estacao.DataGridView1.Rows.Count - 1
            Dim eixox = Estacao.DataGridView1.Item(0, Count).Value
            Dim eixoy = Estacao.DataGridView1.Item(2, Count).Value
            Chart1.Series(0).Points.AddXY(eixox, eixoy / 1)
            Chart1.ResetAutoValues()
        Next
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Me.Close()
    End Sub
End Class
```

A variável umidade foi utilizada como exemplo, porém o processo é igual para todas as outras variáveis, alterando apenas os nomes para as variáveis correspondentes.

O trecho de código mostra que foi criado um contador que vai de zero até a quantidade de linhas da planilha, e coleta para o eixo das abscissas da estampa de tempo e para o eixo das ordenadas o valor da variável desejada, onde o número 1 da linha de código que especifica o Eixo Y, é a coluna da planilha, e é mudado de acordo com a variável desejada, todos estes valores são armazenados em vetores e exibidos pelo gráfico, conforme exibido pelas Figuras 76 e 77.

| Estampa de Tempo | Temperatura (C) |
|---|---|
| 18-11-2015 06:46:30 | 29,86 |
| 18-11-2015 06:47:30 | 29,86 |
| 18-11-2015 06:48:30 | 29,86 |
| 18-11-2015 06:49:31 | 29,86 |
| 18-11-2015 06:50:31 | 29,87 |
| 18-11-2015 06:51:32 | 29,88 |
| 18-11-2015 06:52:32 | 29,89 |
| 18-11-2015 06:53:32 | 29,89 |

Figura 76 – Dados coletados de Temperatura.
Fonte: Autor.

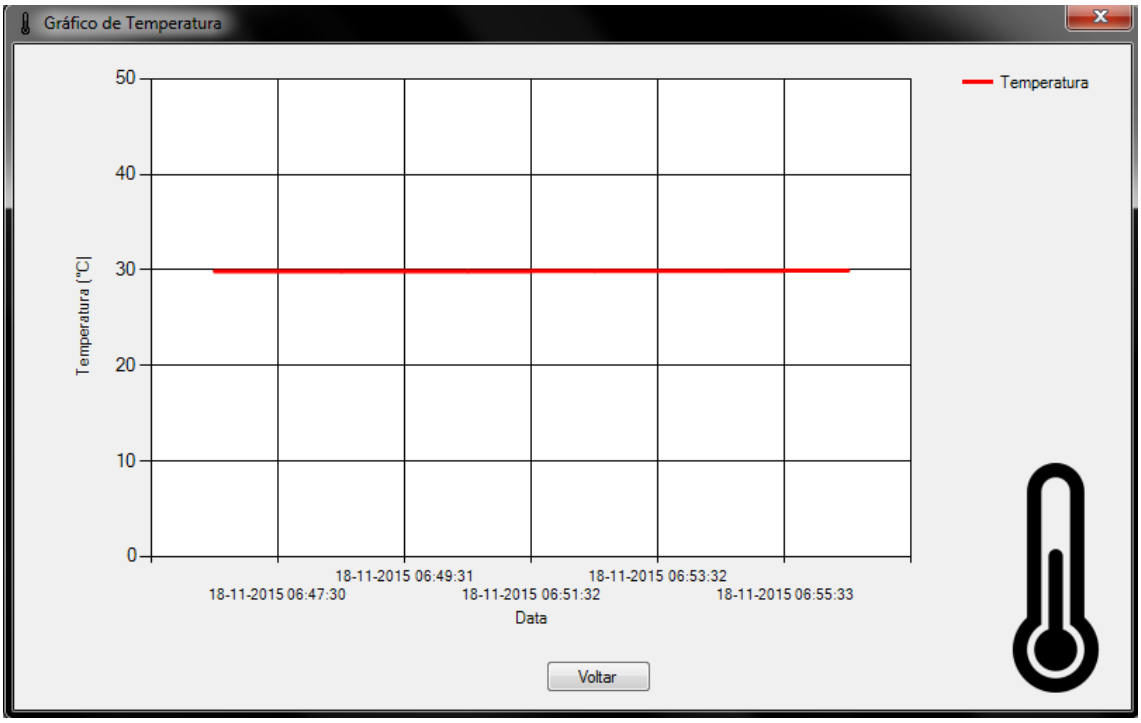



Figura 77 – Exibição gráfica dos dados coletados de temperatura.
Fonte: Autor.

## 7.2. Testes Finais com a Estação meteorológica

Para o teste final do Projeto de Graduação, gostaríamos de testar e comparar os dados colhidos pelo protótipo desenvolvido com alguma estação localizada na cidade. Durante esta pesquisa, foi encontrada uma de propriedade da universidade, porém localizada no pólo de Nova Friburgo, impossibilitando esta comparação local. Tentou-se também contato com estações no CEFET-RJ e no forte de Copacabana, ambas sem sucesso.

Os experimentos foram realizados em Bonsucesso, Rio de Janeiro, residência de um dos granduandos, na data de 29 de novembro de 2015. Neste teste foi avaliada a consistência dos valores coletado pela estação em comparação com os dados coletados de diversas fontes que serão listadas a seguir. Como são analisadas variações ambientais que se modificam muito lentamente ao passar do tempo, não é necessário a aquisição dos dados numa frequência alta, então foi escolhido o intervalo de aquisição de dados de hora em hora por motivos dos sites coletarem os dados com essa frequência. A medição é exibida na Figura 78 e os resultados são exibidos nas Figuras 79 a 84. Para nossa referência, foram coletados valores de quatro fontes diferentes, que são:

- http://www.climatempo.com.br/previsao-do-tempo/cidade/321/riodejaneiro-rj
- http://tempo1.cptec.inpe.br/cidades/tempo/241
- http://br.weather.com/agora/BRRJ0209:1:BR, que aquisita dados de Bonsucesso em um intervalo de 3 em 3 horas.
- http://alertario.rio.rj.gov.br/?page_id=862, que fornece os dados do bairro de São Cristovão.

O teste foi realizado em um intervalo das 12 às 15 horas, e todos os dados aquisitados pela estação projetada serão exibidos a seguir. Os valores coletados nos *web sites* de referência estão no Anexo D.

Figura 78 – Estação Meteorológica em ambiente externo.
Fonte: Autor.

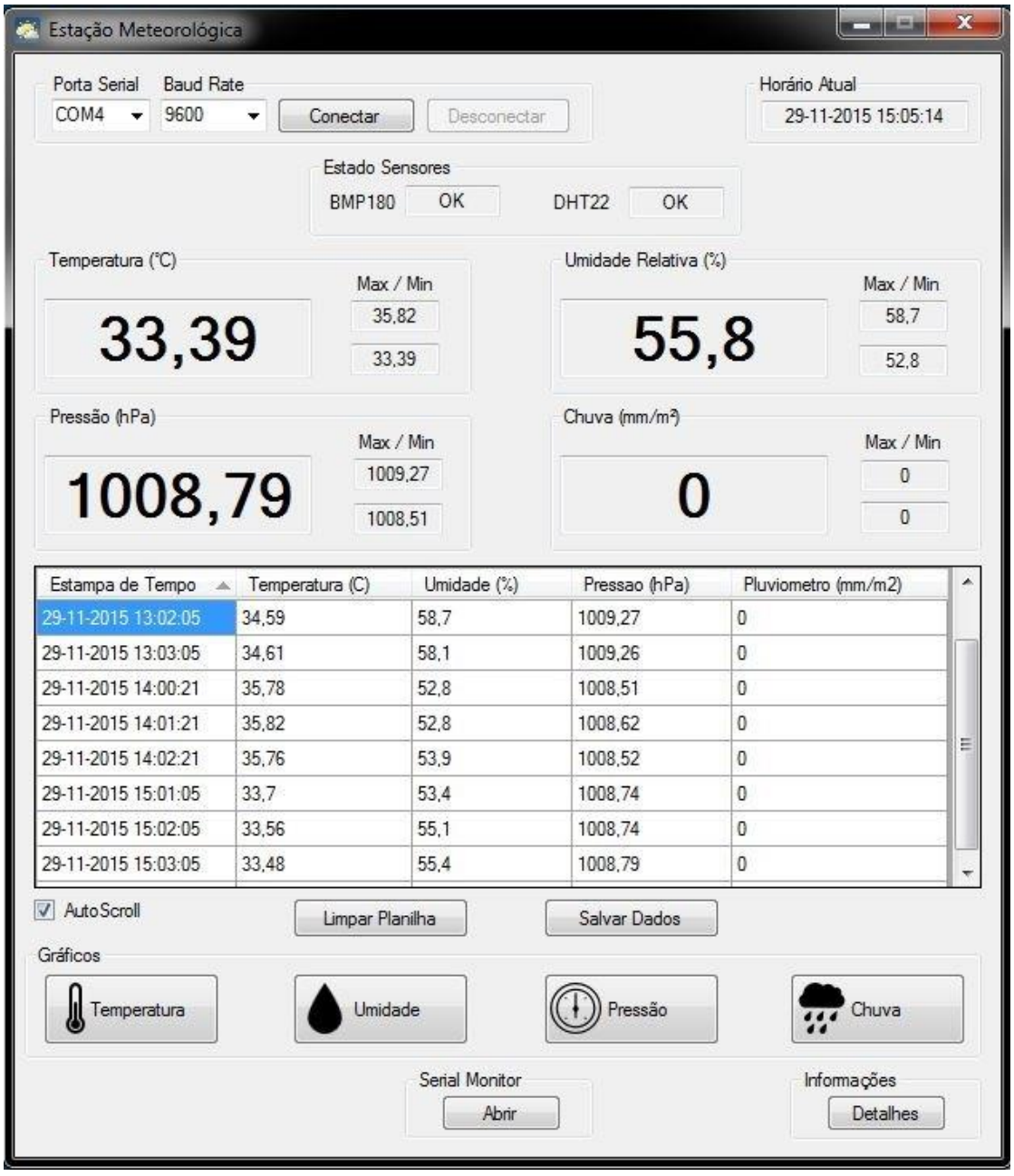



Figura 79 – Dados aquisitados pela estação meteorológica em 29/11/2015.
Fonte: Autor.

| Estampa de Tempo | Temperatura (C) | Umidade (%) | Pressao (hPa) | Pluviometro (mm) |
|---|---|---|---|---|
| 29-11-2015 11:58 | 31,22 | 67,4 | 1009,06 | 0 |
| 29-11-2015 11:59 | 31,2 | 67,5 | 1009,07 | 0 |
| 29-11-2015 12:00 | 31,2 | 67,6 | 1009,07 | 0 |
| 29-11-2015 13:01 | 34,57 | 57,4 | 1009,24 | 0 |
| 29-11-2015 13:02 | 34,59 | 58,7 | 1009,27 | 0 |
| 29-11-2015 13:03 | 34,61 | 58,1 | 1009,26 | 0 |
| 29-11-2015 14:00 | 35,78 | 52,8 | 1008,51 | 0 |
| 29-11-2015 14:01 | 35,82 | 52,8 | 1008,62 | 0 |
| 29-11-2015 14:02 | 35,76 | 53,9 | 1008,52 | 0 |
| 29-11-2015 15:01 | 33,7 | 53,4 | 1008,74 | 0 |
| 29-11-2015 15:02 | 33,56 | 55,1 | 1008,74 | 0 |
| 29-11-2015 15:03 | 33,48 | 55,4 | 1008,79 | 0 |
| 29-11-2015 15:04 | 33,39 | 55,8 | 1008,79 | 0 |

Figura 80 – Dados armazenados no registro criado pelo *software* da Estação Meteorológica.
Fonte: Autor.

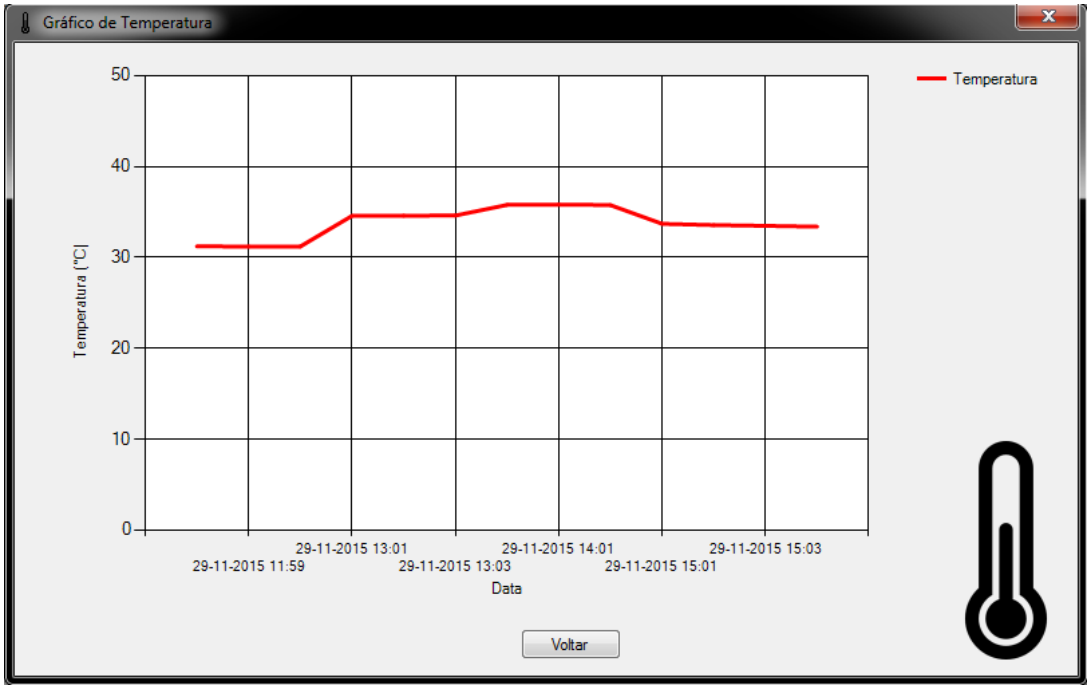



Figura 81 – Gráfico da variação de temperatura durante o período do teste final.
Fonte: Autor.

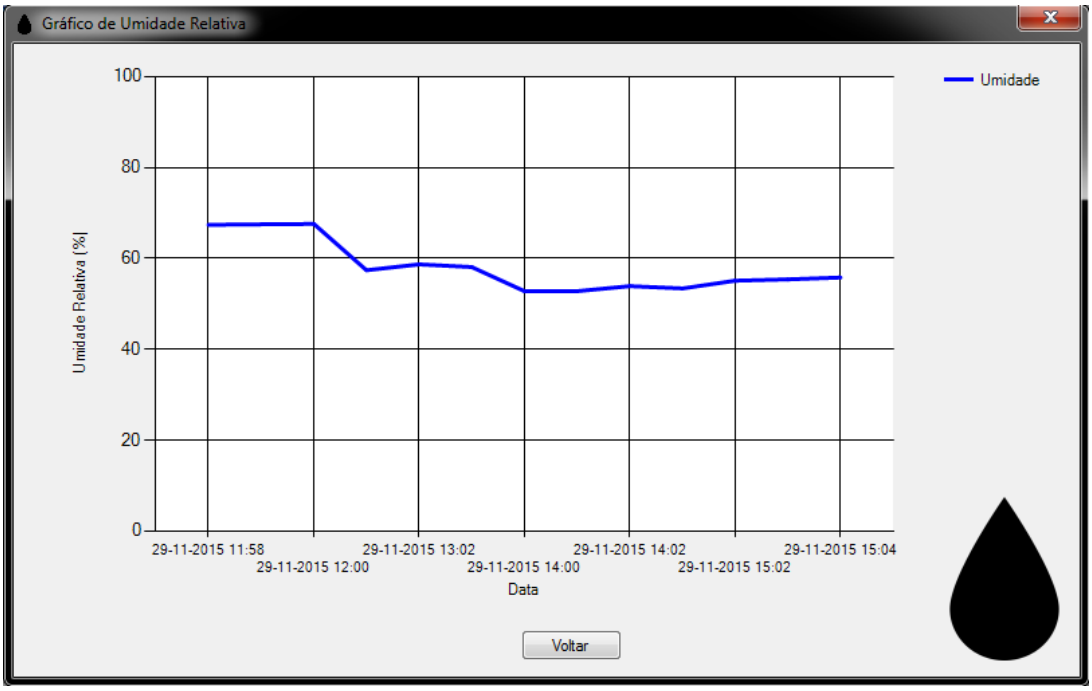



Figura 82 – Gráfico da variação de umidade durante o período do teste final.
Fonte: Autor.

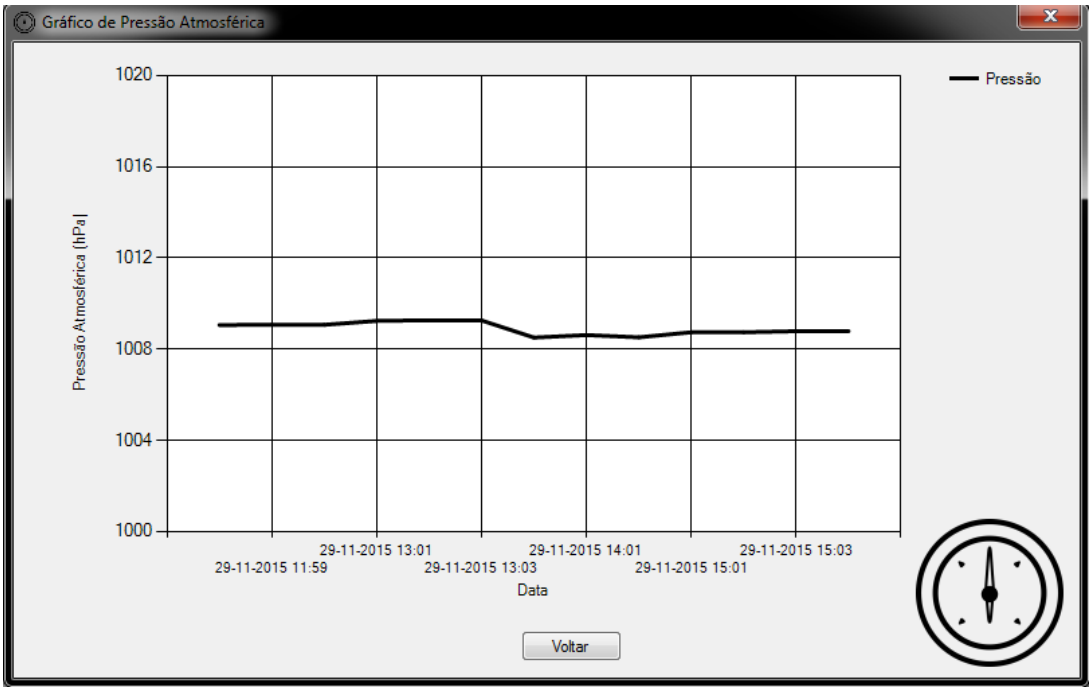



Figura 83 – Gráfico da variação de pressão durante o período do teste final.
Fonte: Autor.

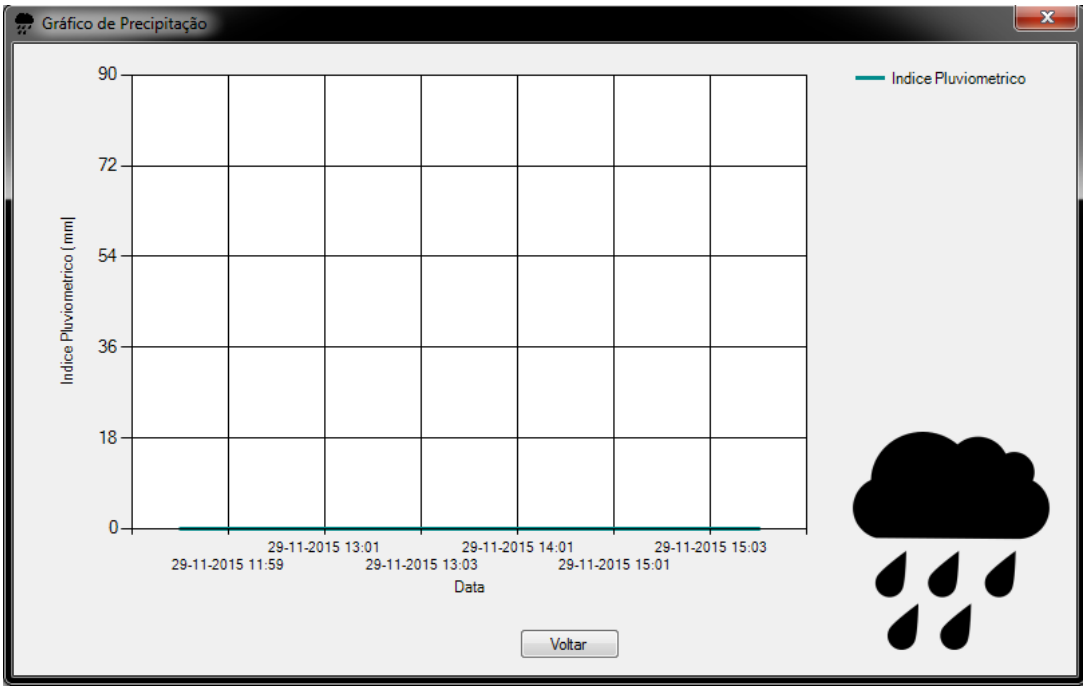



Figura 84 – Gráfico da variação do índice pluviométrico durante o período do teste final.
Fonte: Autor.

### 7.3. Resultados, análise e discussão de dados

O projeto desenvolvido mostrou-se eficiente em relação à transmissão e recepção de dados pelo módulo, que estava aproximadamente 50 metros de distância. Não foi testado em maiores distâncias por impossibilidade da garantia de segurança do protótipo e dos alunos em localidades mais altas, como tetos de prédios, ou em campos abertos, como o morro do Sumaré.

Os dados colhidos pela Estação Meteorológica desenvolvida têm valores semelhantes em relação aos dados das outras estações, com uma diferença razoável esperada por estarem em locais distintos uma em relação à outra. Podemos analisar também que todas as estações possuem sensores diferentes, concluindo que os sensores BMP180 e DHT22 tiveram seus desempenhos satisfatórios comparados as medições das estações de monitoramento localizadas ao longo da cidade. Concluímos que os sensores atenderam a seus propósitos de escopo. Durante o teste, não houve precipitação, mas vimos durante o desenvolvimento do projeto que o pluviômetro tem seu funcionamento conforme o esperado teórico.

O *software* também atendeu a todos os recursos de seu projeto original, desde aquisição de dados até exportação de registro dos mesmos.

Conforme dito no capítulo 3, nas especificações do microcontrolador, suas saídas fornecem no máximo 40 mA por pino e o microprocessador ATmega328 suporta 200 mA máximos para operação. A Tabela 10 mostra o consumo de corrente máxima teórica em funcionamento e em *standby* de cada componente do equipamento para o dimensionamento de carga ao ligarmos com um banco de baterias recarregáveis. A corrente total consumida pelo microcontrolador é a soma das correntes dos sensores.

*Tabela 10 – Consumo de corrente da Estação Meteorológica.*

| **Equipamento** | **Corrente em StandBy (mA)** | **Corrente em Operação (mA)** |
|---|---|---|
| Sensor BMP180 | 0,05 | 1,5 |
| Sensor DHT22 | 0,0001 | 0,005 u |
| Sensor HC-SR04 | 2 | 15 |
| Módulo APC220 | 35 | 35 |
| Total | 37,0501 | 51,505 |

O módulo APC220 não possui modo de *standby* por se tratar de um *transciever*, ou seja, é um receptor e transmissor, assim estando sempre em funcionamento para transmissão e recepção de sinal de dados, onde neste projeto foi usado somente como transmissor, mas não foi concebida a sua operação em *standby*.

Podemos concluir que o maior responsável pelo consumo de corrente da estação meteorológica é o módulo de comunicação APC220. Em operação este módulo representa uma porcentagem menor, mas ainda é o maior consumo. O modo de operação desta estação representa um valor temporal muito pequeno, porque a estação não aquisita dados a todo o momento, onde nos testes só foi realizado medições de hora em hora, então podemos concluir, de acordo com uma média ponderada, de que a corrente média de consumo desta estação é na faixa de 37 mA.

## 8. CONCLUSÃO

Foi projetada uma estação meteorológica que será aplicada a uma embarcação.

Concluímos que o projeto cumpriu o objetivo inicial que foi proposto. Ao final do projeto a estação consegue realizar medições da temperatura ambiente, umidade relativa do ar, pressão atmosférica e presença de chuva. Os dados são obtidos com uma resolução e precisão que estão perfeitamente satisfatórios para a aplicação desejada.

Esses dados são transmitidos para um computador para serem medidos em tempo real, tendo alcance médio de 1000 metros. O *software* permite que estes valores sejam armazenados em um arquivo para análise, contendo data e hora que foram realizadas as medições. Outra funcionalidade desenvolvida para o projeto pelo *software* foi a geração de gráficos no período de tempo determinado pelo usuário com as informações medidas pela estação.

A estação foi projetada de forma que estivesse protegida pelas intempéries existentes, sendo protegida por uma caixa estanque para proteção de todos seus componentes aos fenômenos naturais a qual estará sujeito a embarcação, como forte incidência solar, chuvas, ondas fortes, etc.

Como o consumo da estação é bem baixo, o projeto foi elaborado de forma que a alimentação para toda a estação fosse proveniente da própria embarcação a qual estará embarcada.

A construção do protótipo proposto neste trabalho foi possível graças aos conhecimentos adquiridos durante todo o curso de Engenharia Elétrica, que foram cruciais para construção do projeto. Foram aplicados conceitos de Circuitos Eletrônicos, Controle e Servomecanismos, Eletrônica Analógica e Digital, Linguagem de Programação, Sistemas de Comunicação, dentre outros.

Também foi necessário ir além dos conhecimentos obtidos durante o curso para aprofundamento em conhecimentos mais específicos para construção do projeto que foi exitosamente construído como projeto de conclusão de curso. Nesta parte foi fundamental a colaboração do Orientador José Paulo, auxiliando com os diversos conhecimentos da área.

Além da proposta do Projeto Final de Graduação do aluno botar em prática os conhecimentos obtidos durante o curso de Engenharia, acreditamos que o sistema possa ser utilizado com a finalidade de auxílio a alguma das diversas áreas que a meteorologia pode auxiliar.

## 8.1. Trabalhos Futuros

Como propostas de trabalhos futuros em desenvolvimento de *hardware* podemos citar a adição de novos sensores como Anemômetro, que é encarregado de medir a velocidade do vento juntamente com sua direção. Inclusão de um Luminometro, que é o sensor de luminosidade e a radiação incidente no local. A construção de um sistema de esvaziamento para o pluviomêtro. A implementação de paineis fotovoltáicos juntamente com um banco de baterias para autonomia energética da estação.

Como trabalho futuro de programação, poderia ser desenvolvida a gravação dos dados em um banco de dados SQL (*Structured Query Language*), desenvolvimento de um código onde seja possível acessar estes dados em qualquer momento para manipulá-los do modo desejado, seja para geração de gráficos ou até mesmo para analise. Caso seja adotada alguma implementação de *hardware* listado anteriomente, indicadores em tempo real do banco de baterias, se está recarregando, gráfico em tempo real da direção do vento. Outro aperfeiçoamento seria a inclusão de *layers* que alternam a unidade medida, como, por exemplo, de ºC para ºF, ou de hPa para atm.

## REFERÊNCIAS

[1] Informações sobre meteorologia. Acessado em 29 de agosto.
Disponível em: http://www.inmet.gov.br/portal/index.php?r=home/page&page=tempo

[2] Informações sobre estações meteorológicas. Acessado em 30 de setembro.
Disponível em: http://ainfo.cnptia.embrapa.br/digital/bitstream/CPACT-2009-09/11694/1/artigo-Reisser_estacoes.pdf

[3] Informações sobre meteorologia. Acessado em 30 de agosto.
Disponível em: http://www.inmet.gov.br/html/rede_obs.php

[4] Informações sobre estações meteorológicas. Acessado em 30 de agosto.
Disponível em: http://revistapesquisa.fapesp.br/2012/08/10/boias-ao-mar/

[5] Informções sobre o Arduino. Acessado em 25 de agosto.
Disponível em:http://www.embarcados.com.br/arduino-uno/

[6] Informções sobre o Arduino. Acessado em 25 de agosto.
Disponível em: https://www.arduino.cc/en/Guide/Windows

[7] TENFEN, Cláudio Roberto. Projeto de um Sistema Remoto para Aquisição de Dados Meteorológicos. Projeto - (Graduação em Engenharia de Computação) - Universidade do Vale do Itajaí, São José, 2013.

[8] AYOADE, J, O. Introdução à Climatologia para os Trópicos. 2ª. ED. Rio de Janeiro; Editora Bertrand Brasil S.A., 1988

[9] BISCARO, Guilherme Augusto: Meteorologia Agricola Básica: Série Engenharia Volume I. 2007.

[10] Rashid, M, M; Ferdaus, M. M. "Developmente of Eletronic Rain Gauge System",2015. Disponivel em: http://www.ijeee.net/uploadfile/2014/0807/20140807105927243.pdf

[11] Informações sobre sensores capacitivos. Acessado em 3 de setembro.
Disponível em: http://www.ufrgs.br/eng04030/Aulas/teoria/cap_07/senscapa.htm

[12] Informações sobre sensres piezo resistivos. Acessado em 5 de setembro.
Disponível em: http://www.eletrica.ufpr.br/edu/Sensores/20061/Marcos.html

[13] Informações sobre sensosres ultrassônicos. Acessado em 30 de novembro.
Disponível em: http://www.newtoncbraga.com.br/index.php/como-funciona/5273-art691

[14] PINTO, Paulo Henrique Silva; MOK, Roberto Wu. Projeto de uma Embarcação Multicasco Teleoperada. Projeto (Graduação em Engenharia Elétrica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2015.

[15] Informações sobre modulação GSFK. Acessado em 15 de outubro.
Disponivel em:http://signals.radioscanner.ru/info/item68/

[16] Informações sobre o software Fritzing. Acessado em 22 de setembro.
Disponível em: http://fritzing.org/home/

[17] Informações sobre o sensor BMP180. Acessado em 20 de agosto.
Disponível em:http://cdn.sparkfun.com/datasheets/Sensors/Pressure/BMP180.pdf

[18] Informações sobre o sensor DHT22. Acessado em 21 de agosto. Disponível em:
https://www.adafruit.com/datasheets/Digital%20humidity%20and%20temperature%20sensor%20AM2302.pdf

[19] Informações sobre o sensor HC-SR04. Acessado em 25 de novembro.
Disponível em: http://www.micropik.com/PDF/HCSR04.pdf

[20] Informações sobre o módulo APC220. Acessado em 1 de setembro.
Disponível em: http://www.robotshop.com/media/files/PDF/dfrobot-apc220-manual.pdf

[21]Informações sobre a Caixa Light Steck. Acessado em 25 de agosto.
Disponível em: http://catalogo.steck.com.br/viewitems/caixas-light/caixa-light-154-x-110-x-70?

[22] Informações sobre o software Visual Studio Community. Acessado em 18 de outubro.
Disponível em:https://www.visualstudio.com/products/visual-studio-community-vs

[23] Halvorson, Michael; "*Start Here! Learn Microsoft Visal Basic 2012*"; Microsoft Pres, 2012.

[24] COUTINHO, Tauan;"*Montagem de um Sistema de Aquisição de Dados"*, Projeto de Graduação da faculdade de engenharia – Departamento de Eletrônica e Telecomunicações – Universidade do Estado do Rio de Janeiro,2013.

[25] Informações sobre o conversor USB-TTL CP2102. Acessado em 2 de setembro.
Disponivel em:http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx

[26] Informações sobre os sistemas RAID. Acessado em 2 de setembro.
Disponivel em:http://www.infowester.com/raid.php

[27] Informações sobre o software Serial Port Utility. Acessado em 3 de setembro.
Disponivel em:http://www.darkwood.me/serialport/

[28] Informações sobre a bibilioteca Software Serial do Arduino. Acessado em 4 de setembro. Disponivel em:https://www.arduino.cc/en/Reference/SoftwareSerial

[29] Willis, Thearon; Newsome, Bryan; "*Begginnig Visual Basic 2010*", Wrox, 2010.

[30]Banks, Richard; "*Visual Studio 2012 Cookbook*"*; Packt Publishin, 2012.*

[31] Informações sobre a integração do Arduino com o Visual Studio. Acessado em 20 de outubro. Disponível em: http://www.embarcados.com.br/comunicacao-serial-c-arduino-parte-2/

[32] Informações sobre a integração do Arduino com o Visual Studio. Acessado em 22 de outubro. Disponível em: http://stackoverflow.com/questions/22415743/how-can-i-easily-save-datagridview-to-xls-or-or-csv-without-prompting-user-wit

[33] Informações sobre a integração do Arduino com o Visual Studio. Acessado em 22 de outubro. Disponível em: http://www.theengineeringprojects.com/2012/10/microsoft-visual-basic-2010-com-port-tutorial.html

[34] Informações sobre a integração do Arduino com o Visual Studio. Acessado em 23 de outubro. Disponível em: https://social.msdn.microsoft.com/Forums/pt-BR/home

[35] Informações sobre o Higrômetro. Acessado em 2 de dezembro. Disponível em: http://pt.aliexpress.com/item/Quality-box-violin-guitar-professional-hygrometer-mechanical-wet-disc-humidity-meter/1666251603.html.

[36] Informações sobre os componentes do projeto. Multiplos acessos.
Disponível em: http://labdegaragem.com/

[37] Informações sobre o Termômetro. Acessado em 1 de dezembro.
Disponível em: http://static.hsw.com.br/gif/therm-bulb.gif.

[38] Informações sobre o Termômetro. Acessado em 1 de dezembro.
Disponível em: http://www.eletronpi.com.br/images/fig5trans.png.

[39] Informações sobre o Barômetro. Acessado em 2 de dezembro.
Disponível em:http://www.infoescola.com/wp-content/uploads/2009/08/pressao-atmosferica.jpg

[40] Informações sobre o Barômetro. Acessado em 3 de dezembro.
Disponível em: http://www.estudopratico.com.br/wp-content/uploads/2014/07/o-barometro.jpg

## ANEXO A - DIAGRAMA ELÉTRICO DO SENSOR HC-SR04

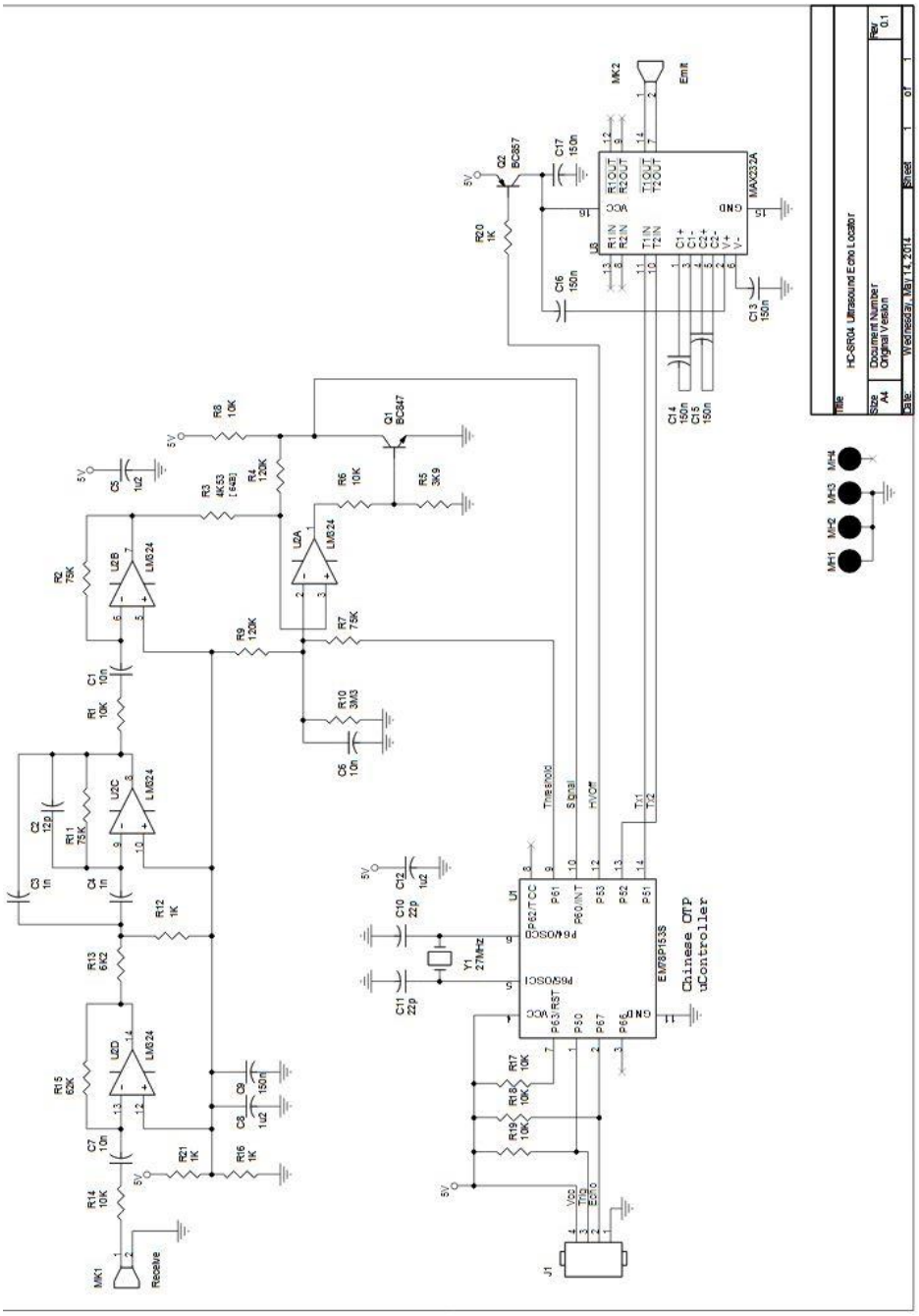



Figura A.1 – Diagrama elétrico do sensor de distância HC-SR04. Extraído de [19].

## ANEXO B – CÓDIGO EXECUTADO ARDUINO

Neste anexo são mostrados os códigos-fonte que utilizamos para o teste e funcionamentodo *hardware* da Estação Meteorológica.

### B.1 – Código da Estação Meteorológica

```
// Inclusão das bibliotecas necessárias
#include <dht.h>
#include <SFE_BMP180.h>
#include <Wire.h>
#include <SoftwareSerial.h>

// Definição da pinagem utilizada
SoftwareSerial apc220(2,3);       //2 =  RX, 3 = TX
#define DHT22_PIN 7
#define sensordechuva 8

// Atribuição de nomes aos sensores
dht DHT;
SFE_BMP180 BMP;

void setup()
{
        Serial.begin(9600);
        apc220.begin(9600);
        pinMode(sensordechuva,INPUT);
}

void loop()
{
// Inicialização e teste dos sensores
        Serial.println("Testando Sensores:");
        Serial.print("BMP: ");
        apc220.println("TestandoSensores:");
        apc220.print("BMP: ");

        if (BMP.begin())
         {
                // 1 = Sensor OK
                Serial.println("1");
                apc220.println("1");
        }
        else
```

```
        {
                //2 = Sensor não encontrado. Checar Fiação
                Serial.println("2");
                apc220.println("2");
        }

        Serial.print("DHT: ");
        apc220.print("DHT: ");
        intchk = DHT.read22(DHT22_PIN);
        switch (chk)
        {
        case DHTLIB_OK:
        //1 = Sensor OK
        Serial.println("1");
        apc220.println("1");
        break;

        case DHTLIB_ERROR_CHECKSUM:
        //2 = Falha na checagem da biblioteca
        Serial.println("2");
        apc220.println("2");
        break;

        case DHTLIB_ERROR_TIMEOUT:
        //3 = Sensor não encontrado
        Serial.println("3");
        apc220.println("3");
        break;
        }

        Serial.println();               //linha para separar os dados
        apc220.println();               //linha para separar os dados

        // Aquisição das medidas de Temperatura e Pressão
        char status;                    // Cria um vetor intermediario
        double T,P;       // Cria variaveis para retorno de valor deste vetor


        status = BMP.startTemperature();          // Inicia a medição de Temperatura
        if (status != 0)
        {
                delay(status);                                  // Delay para aquisição da medida
                status = BMP.getTemperature(T);
// Retorna com a medida na variavel T. A função retona 1 se bem sucedida, 0 para falha.
                if (status != 0)
```

```
                {
                        // Exibe a medida.
                        Serial.print("TEMP = ");
                        Serial.println(T,2);
                        apc220.print("TEMP = ");
                        apc220.println(T,2);

// Inicia a medição de pressão.
    // A medida é feita atraves de sobreamostragem, de 0 a 3. Olhar datasheet. 3 - Resolução Máxima
    // Se o pedido for sucedido, aguardamos o valor parametro como nilissegundos.
    // Se for falho, retorna 0.

                        status = BMP.startPressure(3);     // Inicia a medição de pressão.
                        if (status != 0)
                        {
                                delay(status);                // Delay para aquisição da medida.
                                status = BMP.getPressure(P,T);
// Retorna com a medida na variavel P. Notar que a aquisição também é função de T, caso esta seja estável, o sensor aquisita a medição de Pressão.
                                if (status != 0)   // A função retona 1 se bem sucedida, 0 para falha.
                                {
                                        // Exibe a medida.
                                       Serial.print("PRES = ");
                                        Serial.println(P/1013.2501,2);
                                        apc220.print("PRES = ");
                                        apc220.println(P/1013.2501,2);
                                }
                                else
                                //Falha 1 = Falha no retorno da medida de pressão
                                Serial.println("BMP FALHA 1")&apc220.println("BMP FALHA 1");
                        }
                //Falha 2 = Falha na aquisição da medida de pressão
                Serial.println("BMP FALHA 2")&apc220.println("BMP FALHA 2");
        }
        else
        //Falha 3 = Falha no retorno da medida de temperatura
        Serial.println("BMP FALHA 3")&  apc220.println("BMP FALHA 3");
}
else
  //Falha 4 = Falha na aquisição da medida de temperatura
Serial.println("BMP FALHA 4")&apc220.println("BMP FALHA 4");


//Aquisição e exibição das medidas de Umidade
Serial.print("HUM = ");
```

```
Serial.println(DHT.humidity, 2);
apc220.print("HUM = ");
apc220.println(DHT.humidity, 2);

//Aquisição da informação de precipitação
float cm, dist, chuva;
long microsec = ultrasonic.timing();

cm = ultrasonic.convert(microsec, Ultrasonic::CM);
dist = 13.30; // distancia sem água no recipiente
chuva = (dist - cm)*10; //altura em cm

Serial.print("CHUVA = ");
Serial.println(chuva);
apc220.print("CHUVA = ");
apc220.println(chuva);

Serial.println();          //linha para separar os dados
apc220.println();          //linha para separar os dados


delay(60000);              //1 min para nova aquisiçao
}
```

## B.2 – Código de calibração do Sensor BMP180

```
// Inclusão das bibliotecas necessárias
#include <SFE_BMP180.h>
#include <Wire.h>

// Endereço I2C do BMP180
#define BMP180_ADDRESS 0x77

// OSS = Oversampling Setting = Configuração de sobreamostragem
const unsigned char OSS = 0;

// Valores de Calibração
int ac1;
int ac2;
int ac3;
unsignedint ac4;
unsigned int ac5;
unsignedint ac6;
int b1;
```

```
int b2;
intmb;
int mc;
int md;
long b5;

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

	Bmp180Calibration();
}

void loop()
{
	// Inclusão das variaveis e como elas são aquisitadas
	float temperature = bmp180GetTemperature(bmp180ReadUT());
	float pressure = bmp180GetPressure(bmp180ReadUP());

	// Exibição dos dados
	Serial.print("Temperatura = ");
	Serial.print(temperature, 2);
	Serial.println(" graus Celsius");

	Serial.print("Pressao = ");
	Serial.print(pressure/1013.2501, 2);
	Serial.println(" atm");

	Serial.println();

	delay(5000);
}

// Processo de Calibração
void bmp180Calibration()
{
	ac1 = bmp180ReadInt(0xAA);
	ac2 = bmp180ReadInt(0xAC);
	ac3 = bmp180ReadInt(0xAE);
	ac4 = bmp180ReadInt(0xB0);
	ac5 = bmp180ReadInt(0xB2);
	ac6 = bmp180ReadInt(0xB4);
	b1 = bmp180ReadInt(0xB6);
	b2 = bmp180ReadInt(0xB8);
```

```
        mb = bmp180ReadInt(0xBA);
        mc = bmp180ReadInt(0xBC);
        md = bmp180ReadInt(0xBE);
}


//CalculoTemperatura Real
float bmp180GetTemperature(bmp180ReadUT());
{
        long x1, x2;

        x1 = (((long)ut - (long)ac6)*(long)ac5)/32768;
        x2 = ((long)mc * 2048)/(x1 + md);
        b5 = x1 + x2;

        float temp = ((b5 + 8)/16);
        return temp;
}

//CalculoPressão Real
long bmp180GetPressure(bmp180ReadUP());
{
        long x1, x2, x3, b3, b6, p;
        unsigned long b4, b7;

        b6 = b5 - 4000;

        x1 = (b2 * (b6 * b6)/4096)/2048;
        x2 = (ac2 * b6)/2048;
        x3 = x1 + x2;
        b3 = (((((long)ac1)*4 + x3)<<OSS) + 2)/4;


        x1 = (ac3 * b6)/8192;
        x2 = (b1 * ((b6 * b6)/2048))/65536;
        x3 = ((x1 + x2) + 2)/4;
        b4 = (ac4 * (unsigned long)(x3 + 32768))/32768;

        b7 = ((unsigned long)(up - b3) * (50000>>OSS));
        if (b7 < 0x80000000)
                p = (b7*2)/b4;
        else
                p = (b7/b4)*2;

        x1 = (p/256) * (p/256);
```

```
        x1 = (x1 * 3038)/65536;
        x2 = (-7357 * p)/65536;
        p += (x1 + x2 + 3791)/16;

        longpres = p;
        returnpres;
}
```

## B.3 – Código de calibração do Sensor HC-SR04

```
// Inclusão da biblioteca necessária
#include <Ultrasonic.h>

// Definição da pinagem utilizada
#define TRIGGER_PIN  8
#define ECHO_PIN     9

Ultrasonic ultrasonic(TRIGGER_PIN, ECHO_PIN);

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

void loop()
{
        float cm, dist, chuva;
        long microsec = ultrasonic.timing();

        cm = ultrasonic.convert(microsec, Ultrasonic::CM);
        dist = 13.38;                   // distancia sem água no recipiente em cm
        chuva = (dist - cm)*10;         //altura em mm

        Serial.print("CHUVA = ");
        Serial.println(chuva);
        Serial.print("Distancia padrao sem agua = ");
        Serial.println(cm);

        delay(5000);
}
```

## B.4 – Código de teste do Módulo de Comunicação APC220

```
// Inclusão da biblioteca necessária
#include <SoftwareSerial.h>

// Definição da pinagem utilizada
SoftwareSerial apc220(2,3);       //2 =  RX, 3 = TX

void setup()
{
        Serial.begin(9600);
        apc220.begin(9600);
}

void loop()
{
        // Envio da mensagem de teste
        Serial.println("Serial");
        apc220.println("RF");
        delay(1000);
}
```

## ANEXO C – Código do Software no *Visual Basic*

Neste anexo são mostrados os códigos-fonte de todo o *software* criado para monitoração da Estação Meteorológica. O código é dividido em várias *Windows Form*, que são as janelas do *software*.

### C.1 - Código da WindowsForm: estacao.vb

```
Imports System
Imports System.ComponentModel
Imports System.Threading
Imports System.IO
Imports System.IO.Ports
Imports System.Data
Imports System.Windows.Forms

Public Class Estacao
    Public atuaTemp As Double
    Public atuaHum As Double
    Public atuaPres As Double
    Public atuaChuva As Double
    Public atuaBMP As Double
    Public atuaDHT As Double
    Public atuaTeste As Double
    'Verifica o status da porta serial. True = Aberta; False = Fechada
    Dim comOpen As Boolean

    Private Sub Estacao_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        'Todas as portas conectadas
        Dim comPorts As String() = System.IO.Ports.SerialPort.GetPortNames
        cbPortas.Items.AddRange(comPorts)
        cbPortas.Text = comPorts(0)

        'Valores possiveis de bps aceitos pelo módulo RF
        cbBaud.Items.Add(9600)
        cbBaud.Items.Add(19200)
        cbBaud.Items.Add(1200)
        cbBaud.Items.Add(2400)
        cbBaud.Items.Add(4800)

        TimerHORA.Enabled = True
        TimerCOM.Enabled = True

        'Status Inicial dos Botoes
        CheckBox1.Checked = True
        btDesconectar.Enabled = False
    End Sub

    'Atualiza e adiciona todas as COM diponíveis na lista
    Private Sub TimerCOM_Tick(sender As Object, e As EventArgs) Handles TimerCOM.Tick
        cbPortas.Items.Clear()
```

```
        For Each porta As String In SerialPort.GetPortNames()
            cbPortas.Items.Add(porta)
        Next
    End Sub

    'Exibe a hora do computador
    Private Sub TimerHORA_Tick(sender As Object, e As EventArgs) Handles TimerHORA.Tick
        tbHora.Text = Date.Now.ToString("dd/MM/yyyy HH:mm:ss")
    End Sub

    'Abre a porta serial
    Private Sub btConectar_Click(sender As Object, e As EventArgs) Handles
btConectar.Click
        'Executa a função criada Conectar()
        Conectar()
    End Sub

    'Fecha a porta serial
    Private Sub btDesconectar_Click(sender As Object, e As EventArgs) Handles
btDesconectar.Click
        'Executa a função criada Desconectar()
        Desconectar()
    End Sub

    'Fecha a porta serial antes de fechar o programa
    Private Sub EstacaoMeteorologica_FormClosing_click(sender As Object, e As
FormClosingEventArgs) Handles Me.FormClosing
        Desconectar()
        'Verifica se tem dados para perguntar se deseja salvar
        If DataGridView1.RowCount.Equals(0) Then

        Else
            'Abre janela perguntando se deseja salvar
            If MessageBox.Show("Deseja salvar os dados antes de fechar?", "Estação
Meteorológica", MessageBoxButtons.YesNo) = DialogResult.Yes Then
                salvaCSV(DataGridView1, "Registro__" & DateAndTime.Day(Now()) & "-" &
Month(Now()) & "-" & Year(Now()) & "__" & Hour(Now()) & "h-" & Minute(Now()) & "m.csv")
                MessageBox.Show("Dados salvos como arquivo 'Registro__" &
DateAndTime.Day(Now()) & "-" & Month(Now()) & "-" & Year(Now()) & "__" & Hour(Now()) & "h-
" & Minute(Now()) & "m.csv' na pasta de Debug do aplicativo.")
            End If
        End If
    End Sub

    'Recepçao dos dados
    Private Sub SerialPort1_DataReceived(ByVal sender As System.Object, ByVal e As
System.IO.Ports.SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived
        If comOpen Then
            Dim dados As String = SerialPort1.ReadLine()

            If dados.Contains("TEMP") Then
                dados = dados.Replace("TEMP = ", "")
                If IsNumeric(dados) Then
                    atuaTemp = CDbl(dados / 100)
```

```
                    tbTemp.Invoke(New UpdateTempBoxDelgate(AddressOf UpdateTempBox),
atuaTemp.ToString)
                End If

            ElseIf dados.Contains("HUM") Then
                dados = dados.Replace("HUM = ", "")
                If IsNumeric(dados) Then
                    atuaHum = CDbl(dados / 100)
                    tbHum.Invoke(New UpdateHumBoxDelgate(AddressOf UpdateHumBox),
atuaHum.ToString)
                End If

            ElseIf dados.Contains("PRES") Then
                dados = dados.Replace("PRES = ", "")
                If IsNumeric(dados) Then
                    atuaPres = CDbl(dados / 100)
                    tbPres.Invoke(New UpdatePresBoxDelgate(AddressOf UpdatePresBox),
atuaPres.ToString)
                End If

            ElseIf dados.Contains("CHUVA") Then
                dados = dados.Replace("CHUVA = ", "")
                If IsNumeric(dados) Then
                    atuaChuva = CDbl(dados / 100)
                    tbPres.Invoke(New UpdateChuvaBoxDelgate(AddressOf UpdateChuvaBox),
atuaChuva.ToString)
                End If

            ElseIf dados.Contains("BMP:") Then
                dados = dados.Replace("BMP:", "")
                If IsNumeric(dados) Then
                    If dados.Contains("1") Then
                        atuaBMP = CDbl(dados)
                        tbBMP.Invoke(New BMPtesteOK(AddressOf BMPok), atuaBMP.ToString)
                    ElseIf dados.Contains("2") Then
                        atuaBMP = CDbl(dados)
                        tbBMP.Invoke(New BMPtesteFALHA(AddressOf BMPfalha),
atuaBMP.ToString)
                    End If
                End If

            ElseIf dados.Contains("DHT:") Then
                dados = dados.Replace("DHT:", "")
                If IsNumeric(dados) Then
                    If dados.Contains("1") Then
                        atuaDHT = CDbl(dados)
                        tbDHT.Invoke(New DHTtesteOK(AddressOf DHTok), atuaDHT.ToString)
                    ElseIf dados.Contains("2") Then
                        atuaDHT = CDbl(dados)
                        tbDHT.Invoke(New DHTtesteFALHA1(AddressOf DHTfalha1),
atuaDHT.ToString)
                    ElseIf dados.Contains("3") Then
                        atuaDHT = CDbl(dados)
```

```
                        tbDHT.Invoke(New DHTtesteFALHA2(AddressOf DHTfalha2),
atuaDHT.ToString)
                    End If
                End If
            End If

        End If

    End Sub

    'Atualiza textBox de Status dos Sensores
    Delegate Sub BMPtesteOK(ByVal t As String)
    Public Sub BMPok(ByVal t As String)
        tbBMP.Text = "OK"
    End Sub

    Delegate Sub BMPtesteFALHA(ByVal t As String)
    Public Sub BMPfalha(ByVal t As String)
        tbBMP.Text = "FORA/OUT"
    End Sub

    Delegate Sub DHTtesteOK(ByVal t As String)
    Public Sub DHTok(ByVal t As String)
        tbDHT.Text = "OK"
    End Sub

    Delegate Sub DHTtesteFALHA1(ByVal t As String)
    Public Sub DHTfalha1(ByVal t As String)
        tbDHT.Text = "ERRO BIB"
    End Sub

    Delegate Sub DHTtesteFALHA2(ByVal t As String)
    Public Sub DHTfalha2(ByVal t As String)
        tbDHT.Text = "FORA/OUT"
    End Sub

    Delegate Sub UpdateTempBoxDelgate(ByVal t As String)

    'Atualiza Valor de Temperuta
    Public Sub UpdateTempBox(ByVal t As String)
        tbTemp.Text = t

        'Temperatura máxima
        If tempMax.Text < tbTemp.Text Then
            tempMax.Text = tbTemp.Text
        End If

        'Temperatura mínima
        If tempMin.Text > tbTemp.Text Then
            tempMin.Text = tbTemp.Text
        End If
    End Sub

    Delegate Sub UpdateHumBoxDelgate(ByVal t As String)
```

```
'Atualiza Valor de Humidade
Public Sub UpdateHumBox(ByVal t As String)
    tbHum.Text = t

    'Humidade máxima
    If humiMax.Text < tbHum.Text Then
        humiMax.Text = tbHum.Text
    End If

    'Humidade mínima
    If humiMin.Text > tbHum.Text Then
        humiMin.Text = tbHum.Text
    End If
End Sub

Delegate Sub UpdatePresBoxDelgate(ByVal t As String)

'Atualiza Valor de Pressão
Public Sub UpdatePresBox(ByVal t As String)
    tbPres.Text = t

    'Pressão máxima
    If presMax.Text < tbPres.Text Then
        presMax.Text = tbPres.Text
    End If

    'Pressão mínima
    If presMin.Text > tbPres.Text Then
        presMin.Text = tbPres.Text
    End If

End Sub

Delegate Sub UpdateChuvaBoxDelgate(ByVal t As String)

'Atualiza Valor de Chuva
Public Sub UpdateChuvaBox(ByVal t As String)
    If t < 0 Then
        tbChuva.Text = "0"
    Else
        tbChuva.Text = Decimal.Round(t)
    End If

    'Chuva máxima
    If chuvaMax.Text < tbChuva.Text Then
        chuvaMax.Text = tbChuva.Text
    End If

    'Pressão mínima
    If chuvaMin.Text > tbChuva.Text Then
        chuvaMin.Text = tbChuva.Text
    End If
```

```
        'Atualiza planilha de dados
        Dim linha As Integer = DataGridView1.Rows.Add
        DataGridView1.Rows.Item(linha).Cells(0).Value = tbHora.Text
        DataGridView1.Rows.Item(linha).Cells(1).Value = tbTemp.Text
        DataGridView1.Rows.Item(linha).Cells(2).Value = tbHum.Text
        DataGridView1.Rows.Item(linha).Cells(3).Value = tbPres.Text
        DataGridView1.Rows.Item(linha).Cells(4).Value = tbChuva.Text
        salvaCSV(DataGridView1, "Registro.csv")

        'Marca AutoScroll
        If CheckBox1.Checked = True Then
            DataGridView1.FirstDisplayedScrollingRowIndex = DataGridView1.RowCount - 1
        End If
    End Sub

    'Função criada Desconectar()
    Public Sub Desconectar()
        If comOpen = True Then
            SerialPort1.Close()

            'Reseta os flags e botoes de controle
            comOpen = False
            btDesconectar.Enabled = False
            btConectar.Enabled = True
            cbPortas.Enabled = True
            cbBaud.Enabled = True
            btMonitor.Enabled = True
        End If
    End Sub

    'Função criada Conectar()
    Public Sub Conectar()

        'Propriedades da porta serial tem que estar de acordo com o configurado no par
transmissor/receptor
        With SerialPort1
            .PortName = cbPortas.Text
            .BaudRate = cbBaud.Text
            .Parity = IO.Ports.Parity.None
            .DataBits = 8
            .RtsEnable = False
            'Tempo maximo sem dados = 5 min
            .ReadTimeout = 300000
        End With

        'Concetando-se a porta serial
        Try
            SerialPort1.Open()
            comOpen = SerialPort1.IsOpen
        Catch ex As Exception
            comOpen = False
        End Try

        'Status dos botoes
```

```
        btDesconectar.Enabled = True
        btConectar.Enabled = False
        cbPortas.Enabled = False
        cbBaud.Enabled = False
        btMonitor.Enabled = False
    End Sub

    'Processo de arquivamento dos dados
    Private Sub btSalvar_Click(sender As Object, e As EventArgs) Handles btSalvar.Click
        'Verifica se tem dados
        If DataGridView1.RowCount.Equals(0) Then

        Else
            'Como salva os dados
            salvaCSV(DataGridView1, "Registro__" & DateAndTime.Day(Now()) & "-" &
Month(Now()) & "-" & Year(Now()) & "__" & Hour(Now()) & "h-" & Minute(Now()) & "m.csv")
            MessageBox.Show("Dados salvos como arquivo 'Registro__" &
DateAndTime.Day(Now()) & "-" & Month(Now()) & "-" & Year(Now()) & "__" & Hour(Now()) & "h-
" & Minute(Now()) & "m.csv' na pasta de Debug do aplicativo.", "Estação Meteorológica")
        End If
    End Sub

    'Função salvaCSV criada
    Public Sub salvaCSV(dgv As DataGridView, saidaDado As String)
        If dgv.RowCount > 0 Then
            Dim dado_ent As String = ""
            Dim linha As New DataGridViewRow()
            Dim valor_saida As New StreamWriter(saidaDado)

            'Salva os nomes das colunas no arquivo
            For i As Integer = 0 To dgv.Columns.Count - 1
                If i > 0 Then
                    valor_saida.Write(" " & vbTab & " ") 'espaçamento entre titulos
                End If
                valor_saida.Write(dgv.Columns(i).HeaderText)
            Next

            valor_saida.WriteLine()

            'Salva os dados das tabelas no arquivo
            For j As Integer = 0 To dgv.Rows.Count - 1
                If j > 0 Then
                    valor_saida.WriteLine()
                End If

                linha = dgv.Rows(j)

                For i As Integer = 0 To dgv.Columns.Count - 1
                    If i > 0 Then
                        valor_saida.Write(" " & vbTab & " ") 'espaçamento entre os dados
                    End If

                    dado_ent = linha.Cells(i).Value.ToString()
                    dado_ent = dado_ent.Replace(Environment.NewLine, " ")
```

```
                    valor_saida.Write(dado_ent)
                Next
            Next
            valor_saida.Close()
        End If
    End Sub

    'Apagar linhas da planilha
    Private Sub btDelete_Click(sender As Object, e As EventArgs) Handles btDelete.Click
        'Verifica se tem dados para perguntar se deseja salvar
        If DataGridView1.RowCount.Equals(0) Then

        Else
            'Abre janela perguntando se deseja salvar
            If MessageBox.Show("Deseja salvar os dados antes de limpar a planilha?",
"Estação Meteorológica", MessageBoxButtons.YesNo) = DialogResult.Yes Then
                salvaCSV(DataGridView1, "Registro__" & DateAndTime.Day(Now()) & "-" &
Month(Now()) & "-" & Year(Now()) & "__" & Hour(Now()) & "h-" & Minute(Now()) & "m.csv")
                MessageBox.Show("Dados salvos como arquivo 'Registro__" &
DateAndTime.Day(Now()) & "-" & Month(Now()) & "-" & Year(Now()) & "__" & Hour(Now()) & "h-
" & Minute(Now()) & "m.csv' na pasta de Debug do aplicativo.")
            End If
        End If
        DataGridView1.Rows.Clear()
    End Sub

    'Mostra as informações
    Private Sub btInfo_Click(sender As Object, e As EventArgs) Handles btInfo.Click
        Info.Show()
    End Sub

    'Exibe a janela de gráfico de temperatura
    Private Sub btGrafTemp_Click(sender As Object, e As EventArgs) Handles
btGrafTemp.Click
        tempGraf.Show()
    End Sub

    'Exibe a janela de gráfico de umidade
    Private Sub btGrafHum_Click(sender As Object, e As EventArgs) Handles btGrafHum.Click
        humiGraf.Show()
    End Sub

    'Exibe a janela de gráfico de pressão
    Private Sub btGrafPres_Click(sender As Object, e As EventArgs) Handles
btGrafPres.Click
        presGraf.Show()
    End Sub

    'Exibe a janela de gráfico de chuva
    Private Sub btGrafChuva_Click(sender As Object, e As EventArgs) Handles
btGrafChuva.Click
        chuvaGraf.Show()
    End Sub
```

```
    Private Sub btMonitor_Click(sender As Object, e As EventArgs) Handles btMonitor.Click
        monitor.Show()
    End Sub
End Class
```

## C.2 - Código da WindowsForm: chuvaGraf.vb

```
Public Class chuvaGraf
    Private Sub chuvaGraf_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        For Count As Integer = 0 To Estacao.DataGridView1.Rows.Count - 1
            Dim eixox = Estacao.DataGridView1.Item(0, Count).Value
            Dim eixoy = Estacao.DataGridView1.Item(4, Count).Value
            Chart1.Series(0).Points.AddXY(eixox, eixoy / 1)
            Chart1.ResetAutoValues()
        Next
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Me.Close()
    End Sub
End Class
```

## C.3 - Código da WindowsForm: humiGraf.vb

```
Public Class humiGraf
    Private Sub humiGraf_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        For Count As Integer = 0 To Estacao.DataGridView1.Rows.Count - 1
            Dim eixox = Estacao.DataGridView1.Item(0, Count).Value
            Dim eixoy = Estacao.DataGridView1.Item(2, Count).Value
            Chart1.Series(0).Points.AddXY(eixox, eixoy / 1)
            Chart1.ResetAutoValues()
        Next
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Me.Close()
    End Sub
End Class
```

## C.4 - Código da WindonsForm: presGraf.vb

```
Public Class presGraf
    Private Sub presGraf_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        For Count As Integer = 0 To Estacao.DataGridView1.Rows.Count - 1
            Dim eixox = Estacao.DataGridView1.Item(0, Count).Value
            Dim eixoy = Estacao.DataGridView1.Item(3, Count).Value
            Chart1.Series(0).Points.AddXY(eixox, eixoy / 1)
            Chart1.ResetAutoValues()
        Next
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
```

```
        Me.Close()
    End Sub
End Class
```

## C.5 - Código da WindonsForm: tempGraf.vb

```
Public Class tempGraf
    Private Sub tempGraf_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        For Count As Integer = 0 To Estacao.DataGridView1.Rows.Count - 1
            Dim eixox = Estacao.DataGridView1.Item(0, Count).Value
            Dim eixoy = Estacao.DataGridView1.Item(1, Count).Value
            Chart1.Series(0).Points.AddXY(eixox, eixoy / 1)
            Chart1.ResetAutoValues()
        Next
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Me.Close()
    End Sub
End Class
```

## C.6 - Código da WindonsForm: serialMon.vb

```
Public Class monitor
    Delegate Sub TextoRecebido(ByVal [text] As String)
    Dim portaOpen As Boolean

    Private Sub monitor_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim myPort As String() = System.IO.Ports.SerialPort.GetPortNames 'Todas as portas
conectadas
        cmbPort.Items.AddRange(myPort)
        cmbPort.Text = myPort(0)

        cmbBaud.Items.Add(9600)
        cmbBaud.Items.Add(19200)
        cmbBaud.Items.Add(4800)
        cmbBaud.Items.Add(2400)
        cmbBaud.Items.Add(1200)
        btnDesconectar.Enabled = False
        TimerCOM.Enabled = True
        Estacao.btConectar.Enabled = False
    End Sub


    'Atualiza e adiciona todas as COM diponíveis na lista
    Private Sub TimerCOM_Tick(sender As Object, e As EventArgs) Handles TimerCOM.Tick
        cmbPort.Items.Clear()
        For Each porta As String In SerialPort.GetPortNames()
            cmbPort.Items.Add(porta)
        Next
    End Sub
```

```
    'Botão Conectar
    Private Sub btnConectar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnConectar.Click
        With SerialPort
            .PortName = cmbPort.Text
            .BaudRate = cmbBaud.Text
            .Parity = IO.Ports.Parity.None
            .DataBits = 8
            .RtsEnable = False
            .ReadTimeout = 300000                    'Tempo maximo sem dados = 5 min
        End With

        'Concetando-se a porta serial
        Try
            SerialPort.Open()
            portaOpen = SerialPort.IsOpen
        Catch ex As Exception
            portaOpen = False
        End Try

        'Reseta os flags e botoes de controle
        btnConectar.Enabled = False
        btnDesconectar.Enabled = True
        cmbPort.Enabled = False
        cmbBaud.Enabled = False
    End Sub

    'Botão Desconectar
    Private Sub btnDesconectar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDesconectar.Click
        If portaOpen = True Then
            SerialPort.Close()

            'Reseta os flags e botoes de controle
            btnConectar.Enabled = True
            btnDesconectar.Enabled = False
            cmbPort.Enabled = True
            cmbBaud.Enabled = True
        End If
    End Sub


    'Trecho de recepção dos dados pela porta serial
    Private Sub SerialPort_DataReceived(ByVal sender As Object, ByVal e As
System.IO.Ports.SerialDataReceivedEventArgs) Handles SerialPort.DataReceived
        'Chamada a Função Dados Recebidos criada
        DadosRecebidos(SerialPort.ReadExisting())
    End Sub

    'Função Dados Recebidos
    Private Sub DadosRecebidos(ByVal [text] As String)
        If Me.rtbRecebido.InvokeRequired Then
            Dim dados As New TextoRecebido(AddressOf DadosRecebidos)
            Me.Invoke(dados, New Object() {(text)})
```

```
        Else
            Me.rtbRecebido.Text &= [text]
        End If
    End Sub

    'Exibe sempre a ultima linha como principal
    Private Sub rtbRecebido_TextChanged(sender As Object, e As EventArgs) Handles
rtbRecebido.TextChanged
        rtbRecebido.SelectionStart = rtbRecebido.Text.Length
        rtbRecebido.ScrollToCaret()
    End Sub

    'Fecha a porta serial antes de fechar a janela
    Private Sub monitor_FormClosing(sender As Object, e As FormClosingEventArgs) Handles
Me.FormClosing
        If portaOpen = True Then
            SerialPort.Close()
        End If
        Estacao.btConectar.Enabled = True
    End Sub

    'Botão Voltar
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        If portaOpen = True Then
            SerialPort.Close()
        End If
        Me.Close()
        Estacao.btConectar.Enabled = True
    End Sub
End Class
```

## ANEXO D – REGISTRO DO TESTE FINAL

Neste anexo são mostrados os dados coletados no teste final comparativo do protótipo desenvolvido ao longo deste projeto junto às referencias citadas no texto do projeto.

As referencias foram os sites do AlertaRio, que possui estações espalhadas ao redor da cidade, do site BrWeather, que possui informações locais do bairro, do site do Instituto Nacional de Pesquisas Espaciais (INPE), que informa as medidas da cidade do Rio de Janeiro, não informando onde sua estação fica localizada, e do site ClimaTempo, que também informa as medidas da cidade. Todos os valores usados no estudo são exibidos a seguir.

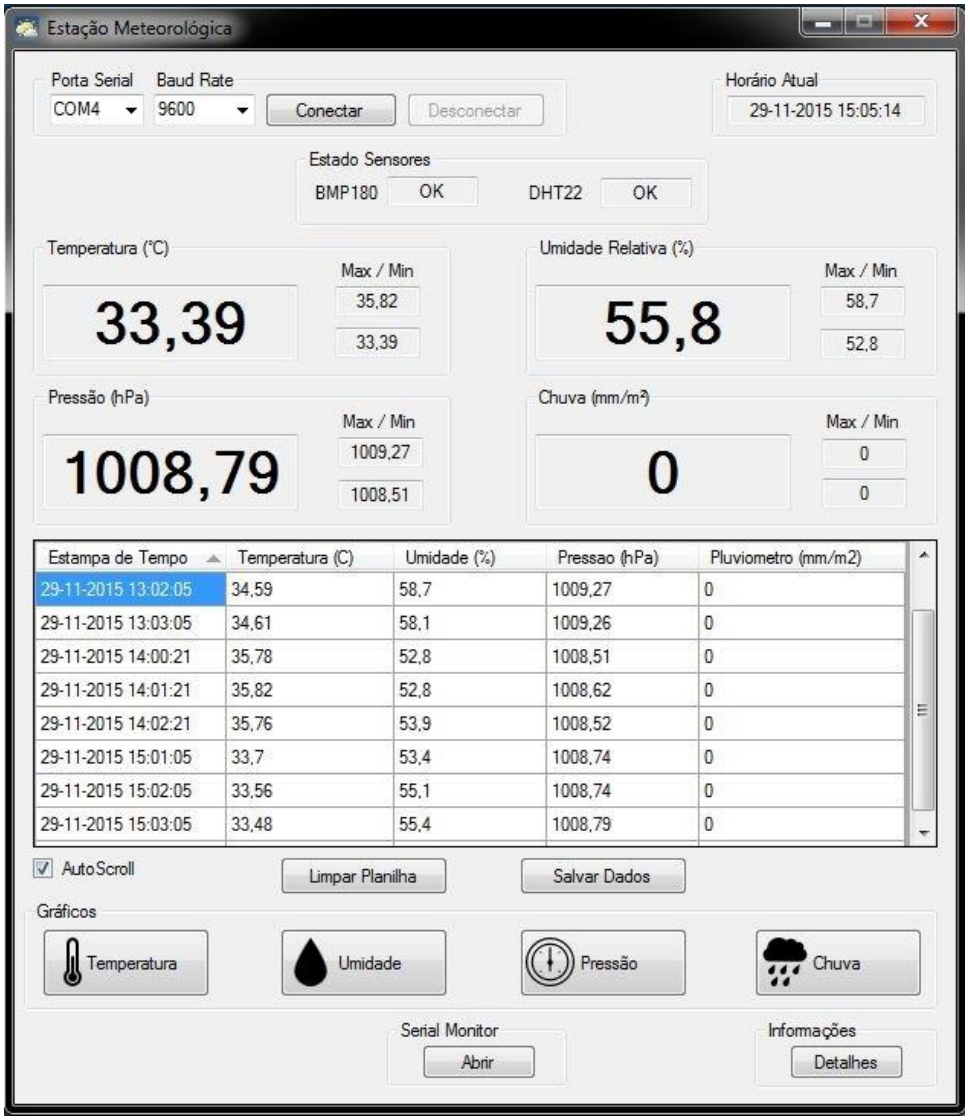



Figura D.1 – Estação Meteorológica das 13 às 15 horas.

Última Atualização: 12:22 - 29/11/2015 - Horário Brasileiro de Verão

| Relatório Meteorológico da Estação São Cristóvão | | | | | | |
|---|---|---|---|---|---|---|
| Dia | Hora | Direção Vento (graus) | Velocidade Vento (Km/h) | Temperatura (celsius) | Pressão (hPa) | Umidade (%) |
| 29/11/2015 | 12:15:00 | - | 0,0 | 32,6 | 1006,1 | 56 |
| 29/11/2015 | 12:00:00 | 096 | 8,1 | 31,3 | 1006,2 | 57 |

Figura D.2 – Dados do site AlertaRio informando as medidas de São Cristóvão às 12 horas.


Figura D.3 – Dados do site BrWeather informando as medidas de Bonsucesso às 12 horas.


Figura D.4 – Dados do site INPE informando as medidas do Rio de Janeiro às 12 horas.


Figura D.5 – Dados do site ClimaTempo informando as medidas do Rio de Janeiro às 12 horas.

Última Atualização: 13:04 - 29/11/2015 - Horário Brasileiro de Verão

| Relatório Meteorológico da Estação São Cristóvão | | | | | | |
|---|---|---|---|---|---|---|
| Dia | Hora | Direção Vento (graus) | Velocidade Vento (Km/h) | Temperatura (celsius) | Pressão (hPa) | Umidade (%) |
| 29/11/2015 | 13:00:00 | 006 | 0,4 | 34,2 | 1005,5 | 49 |

Figura D.6 – Dados do site AlertaRio informando as medidas de São Cristóvão às 13 horas.


Figura D.7 – Dados do site BrWeather informando as medidas de Bonsucesso às 13 horas.


Figura D.8 – Dados do site INPE informando as medidas do Rio de Janeiro às 13 horas.

Última Atualização: 14:02 - 29/11/2015 - Horário Brasileiro de Verão

| Relatório Meteorológico da Estação São Cristóvão | | | | | | |
|---|---|---|---|---|---|---|
| Dia | Hora | Direção Vento (graus) | Velocidade Vento (Km/h) | Temperatura (celsius) | Pressão (hPa) | Umidade (%) |
| 29/11/2015 | 14:00:00 | 126 | 7,4 | 32,8 | 1005,3 | 52 |

Figura D.9 – Dados do site AlertaRio informando as medidas de São Cristóvão às 14 horas.

Rio de Janeiro-RJ


Figura D.10 – Dados do site INPE informando as medidas do Rio de Janeiro às 14 horas.

Última Atualização: 15:04 - 29/11/2015 - Horário Brasileiro de Verão

| Relatório Meteorológico da Estação São Cristóvão | | | | | | |
|---|---|---|---|---|---|---|
| Dia | Hora | Direção Vento (graus) | Velocidade Vento (Km/h) | Temperatura (celsius) | Pressão (hPa) | Umidade (%) |
| 29/11/2015 | 15:00:00 | 324 | 6,4 | 32,0 | 1005,0 | 54 |

Figura D.11 – Dados do site AlertaRio informando as medidas de São Cristóvão às 15 horas.


Figura D.12 – Dados do site INPE informando as medidas do Rio de Janeiro às 15 horas.


Figura D.13 – Dados do site ClimaTempo informando as medidas do Rio de Janeiro às 15 horas.