CENTRO UNIVERSITÁRIO DE BRASÍLIA- UniCEUB

FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS – FATECS

CURSO DE ENGENHARIA DA COMPUTAÇÃO

JOÃO PAULO ALVES SILVA E ROSITO

ESTAÇÃO INTELIGENTE DE COLETA E TRANSMISSÃO DE DADOS METEOROLÓGICOS EM ÁREAS DE RISCO

BRASÍLIA – DF

2° SEMESTRE DE 2014

JOÃO PAULO ALVES SILVA E ROSITO

ESTAÇÃO INTELIGENTE DE COLETA E TRANSMISSÃO DE DADOS METEOROLÓGICOS EM ÁREAS DE RISCO

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Prof. MSc. Luciano Henrique Duque

Brasília

Novembro, 2014

JOÃO PAULO ALVES SILVA E ROSITO

ESTAÇÃO INTELIGENTE DE COLETA E TRANSMISSÃO DE DADOS METEOROLÓGICOS EM ÁREAS DE RISCO

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientador: Prof. MsC. Luciano Henrique Duque.

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas -FATECS.

_____

Prof. Abiezer Amarilia Fernandes
Coordenador do Curso

Banca Examinadora:

_____

Prof. Orientador: Luciano Henrique Duque
Mestre, UniCEUB

_____

Profa. Ingrid Maria Dittert
Doutora, UniCEUB

_____

Prof. Miguel Eduardo Ordoñez Mosquera
Mestre, UniCEUB

Dedico este trabalho à minha família, que me apoiou durante todo o curso de graduação, me educou e me ensinou todos os valores que tenho. Dedico também aos meus amigos, que estiveram sempre ao meu lado e me apoiaram em todos os momentos da minha vida.

João Paulo Alves Silva e Rosito

AGRADECIMENTOS

Agradeço à minha namorada Bianca Tanure pelo apoio prestado em todas as minhas decisões durante a elaboração deste trabalho, pelo amor e pela paciência, que foi fundamental para superação dos momentos difíceis. Ao meu orientador Luciano Duque, professor mestre do UniCEUB, que contribuiu de forma significativa para o desenvolvimento do meu projeto.

Muito obrigado!

João Paulo Alves Silva e Rosito

“Conhecimento não é aquilo que você sabe, mas o que você faz com aquilo que você sabe.”

Aldous Huxley

SUMÁRIO

LISTA DE FIGURA**S**

LISTA DE *QUADROS*

LISTA DE CÓDIGOS

LISTA DE GRÁFICOS

LISTA DE SIGLAS E ABREVIATURAS

RF – Radio Frequency

LCD – Liquid Crystal *Display* (Visor de Cristal Líquido)

INMET – Instituto Nacional de Meteorologia

V – Volts

UR – Umidade Relativa

RESUMO

A variedade dos meios de comunicação que temos hoje, e ainda, a necessidade de informação a todo instante e em qualquer lugar, exigem tecnologias de transmissão de dados que apresentem uma boa relação de custo-benefício e de forma rápida. Este projeto propõe o estudo e desenvolvimento de uma estação meteorológica remota automatizada. O objetivo é tornar possível e eficiente o fornecimento de informações meteorológicas para o atendimento de comunidades carentes e regiões com altos índices de catástrofes naturais, como chuvas fortes que possam causar possíveis deslizamentos de terras. O sistema permitirá que sejam tomadas ações antecipadas de alerta e evacuação dos locais monitorados. Tal solução será implementada com a utilização da tecnologia RF *Link* (transmissão por radiofrequência), aliada a um microcontrolador, no qual serão construídas duas estações, uma de coleta e transmissão das informações, como, temperatura ambiente, umidade relativa do ar, altitude e pressão atmosférica, e outra de recepção e monitoramento das mesmas.

Palavras chaves: Estação meteorológica, Radiofrequência, estação remota automatizada, RF *Link*, Arduino.

ABSTRACT

A variety of forms of communication that we have today, need for information at any time and anywhere, require data transmission technologies that are fast and cost effective. This project proposes a study and development of a remote automated cost effective weather station. The goal is to provide feasible and effective weather information for low income communities and regions with indices of natural disasters, such as heavy rains that can cause landslides. This system allows early actions and evacuation on monitored sites. That solution is to be implemented using RF *Link* (streaming radio frequency) that combined with a micro-controller, will collect and transmit information. The one that collects will gather current temperature, humidity, altitude and atmospheric pressure as for the other station will receive and monitor all of these information.

Keywords: Weather Station, Radio Frequency, Automated remote station, RF *Link*, Arduino.

CAPÍTULO 1 INTRODUÇÃO

No Brasil temos diversos obstáculos que impedem que o país desenvolva sistemas eficazes de prevenção a desastres naturais, tais como enchentes e deslizamentos de terras, em favelas e morros habitados. São exemplos: a falta de vontade política, interesses econômicos, incapacidade dos municípios e o sistema burocrático do país. A urbanização no país ocorreu de forma intensa e desigual. As oportunidades de acesso a qualidade de vida nas cidades não é uma realidade para todos, o que leva grande parte da população menos favorecida financeiramente a ocupar áreas impróprias para moradia, que oferecem riscos a vida. (CARVALHO & GALVAO, 2006)

O trabalho tem como tema um estudo de viabilidade e construção de uma estação meteorológica automatizada. O estudo envolve a integração do conhecimento de meteorologia e engenharia de computação, focada na área de automação, analisando as aplicações deste sistema para regiões que tenham riscos de enchentes e deslizamentos de terra causadas por fenômenos naturais, como grandes chuvas.

É utilizado como base para o desenvolvimento do projeto o *hardware* Arduino, pois é um microcontrolador de baixo custo e de código fonte livre (Open Source), além de ser de fácil programação, pois existem várias bibliografias disponíveis em livros e internet. A maior vantagem do Arduino sobre outras plataformas de desenvolvimento de microcontroladores é a facilidade de sua utilização; pessoas que não são da área técnica podem, rapidamente, aprender o básico e criar seus próprios projetos em um intervalo de tempo relativamente curto. Há uma grande comunidade de pessoas utilizando Arduinos, compartilhando seus códigos e diagramas de circuito para que outros os copiem e modifiquem. A maioria dessa comunidade também está muito disposta a auxiliar outros desenvolvedores. (MCROBERTS, 2011)

Rede sem fio é um meio de comunicação no qual os dados são transmitidos pelo ar atmosférico, por ondas eletromagnéticas, substituindo os meios

físicos. Muitos sistemas de automação com tecnologia *wireless* (sem fio) usam ondas de radiofrequência, ou infravermelho. Para o desenvolvimento do protótipo foram utilizados um módulo de radiofrequência, tornando possível a comunicação entre a base transmissora de dados meteorológicos e a base de monitoramento das informações, sensores de pressão, temperatura, umidade relativa do ar e altitude.

A combinação dos conhecimentos meteorológicos adquiridos após as pesquisas e os benefícios da automação de sistemas de controle, nos permite monitorar e fornecer dados meteorológicos que auxiliem na previsão de variações climáticas tornando possível a tomada de medidas preventivas de catástrofes sociais, com o foco em regiões como favelas e loteamentos irregulares instalados nas encostas de morros urbanos em baixadas junto às margens de cursos d'água.

1.1 Objetivos do trabalho

Objetivo Geral – Desenvolver e implementar uma solução que forneça dados meteorológicos que auxiliem meteorologistas na previsão de possíveis chuvas fortes em regiões com altos índices de deslizamentos de terras e enchentes, possibilitando a tomada de ações antecipadas de alerta e evacuação da comunidade, com objetivo de evitar catástrofes sociais;

Objetivos específicos:

1. Realizar coleta e estudo de bibliografias que serão utilizadas para o desenvolvimento do projeto;
2. Revisão bibliográfica sobre estações meteorológicas automatizadas, sensores e microcontrolador;
3. Desenvolvimento do *software* para estação de coleta das informações meteorológicas e envio das mesmas via RF (Radiofrequência);
4. Desenvolvimento do *software* para estação receptora, que será capaz de receber as informações e mostrar para o usuário, a fim de tornar possível o monitoramento das mesmas;
5. Coletar informações meteorológicas (Temperatura, pressão, umidade e altitude) com o uso dos sensores;

6. Transmitir dados de uma estação para a outra através da tecnologia de RF (Radiofrequência);
7. Receber dados transmitidos em uma estação base e mostrar informações para o usuário em um *display* LCD e *interface web*;
8. Realização de testes dos protótipos a fim de se conseguir obter os dados meteorológicos que auxiliarão meteorologistas na previsão das variações climáticas;

1.2 Motivação

Este projeto tem como motivação a construção de um sistema capaz de monitorar variações meteorológicas, com objetivo de auxiliar na previsão de fortes chuvas, tornando possível antecipar medidas de segurança, evitando catástrofes sociais em regiões que não tem estrutura adequada para resistir a graves alterações climáticas.

O intuito é, além da elaboração do sistema, construir protótipos eficientes, pois o objetivo inicial é instalar em regiões com carência de recursos e infraestrutura adequada, como favelas e morros habitados.

1.3 Metodologia

O escopo do projeto abrange a construção de uma estação meteorológica automatizada com transmissão de informações de maneira remota, com o uso da tecnologia RF (Radiofrequência). No projeto serão contemplados dois polos, uma estação transmissora, que irá coletar os dados meteorológicos através de sensores nela instalados, e uma estação receptora, que servirá como base de recepção e monitoramento dos dados transmitidos.

A primeira etapa é formada pela coleta e estudo de todas as referências bibliográficas usadas para o desenvolvimento do projeto, sendo elas, sobre Arduino Uno, módulo de radiofrequência, estações meteorológicas, linguagem de programação *Wiring* (linguagem baseada em C/C++), sensores barométricos e sensores de umidade relativa do ar;

A segunda etapa é formada pelo desenvolvimento do *hardware* tanto da estação transmissora, quanto da estação receptora. Nessa etapa serão realizadas todas as ligações dos circuitos eletrônicos e sensores com o microcontrolador Arduino Uno, para a estação transmissora, e ligação de todos os circuitos eletrônicos, *display* LCD com o Arduino Uno e instalação deste sistema em rede local para a estação receptora;

A terceira etapa consiste em desenvolver o *software* para a estação transmissora, que é a estação que será instalada nas regiões a serem monitoradas. Esta terá que ser capaz de fazer a leitura dos dados meteorológicos através dos sensores nela instalados, processar as informações no microcontrolador e enviar via RF (radiofrequência) para a estação base (receptora). Ainda nesta etapa será desenvolvido o *software* para a estação receptora, capaz de receber as informações e expor para o usuário em um *display* LCD, e *interface web* possibilitando o monitoramento dos dados;

A quarta etapa será composta pela integração do *software* com o *hardware*, onde será passado o código desenvolvido para o microcontrolador, dando início a fase de testes do protótipo do projeto. Nesta fase serão feitos os ajustes necessários até que o sistema atenda a todos os requisitos pré-estabelecidos.

1.4 Resultados Esperados

Espera-se com este projeto, construir uma estação meteorológica automatizada remota com dois polos (transmissão e recepção dos dados), por meio do desenvolvimento de um sistema baseado em Arduino, que consiga coletar os dados de forma rápida e precisa, através dos sensores nele instalados, e enviar via

radiofrequência para a estação base, que por sua vez, irá receber esses dados e mostrar em um *display* LCD e *interface web* para o usuário.

Construir um sistema de fácil instalação para que seja possível a aquisição em comunidades carentes que não possuem estruturas adequadas de prevenção a catástrofes naturais.

1.5 Organização

Segue descrição para o restante deste trabalho:

O Capítulo 2 apresenta o referencial teórico dos sensores, Arduino, módulo de radiofrequência, linguagem de programação e estação meteorológica. O principal objetivo é fornecer fundamentos teóricos para um entendimento pleno das variáveis que serão fundamentais no desenvolvimento do projeto;

O Capítulo 3 apresenta a solução proposta, mostrando o desenvolvimento do projeto e ilustrando as implementações;

O Capítulo 4 apresenta a análise dos resultados obtidos e aplicação da solução proposta, ilustrando todos os testes realizados durante a construção do protótipo;

O Capítulo 5 tem por objetivo apresentar as conclusões obtidas após o término do desenvolvimento de todo o projeto, seguido de sugestões para futuras aplicações e melhorias do projeto;

CAPÍTULO 2 REFERENCIALTEÓRICO

Neste capítulo serão apresentados os resultados de todas as pesquisas feitas que serviram como bases teóricas para a resolução do problema apresentado e desenvolvimento do protótipo.

2.1 Meteorologia

A meteorologia é uma das ciências que estudam a atmosfera terrestre, que tem como foco o estudo dos processos atmosféricos e a previsão do tempo. O foco de estudo da meteorologia é a investigação dos fenômenos observáveis relacionados com a atmosfera. Os fenômenos meteorológicos estão relacionados com variáveis que existem na atmosfera, que são principalmente a temperatura, a pressão atmosférica e a umidade do ar, suas relações e as suas variações com o passar do tempo, ou seja, ela nos fornece uma visão mais simples das condições atmosféricas que ocorrem em nosso dia a dia, como o vento, chuva, insolação. Assim, é possível entender e prever o tempo nas diversas regiões do planeta. Sua atividade é fundamental para a sociedade, pois com essas informações é possível prever desastres naturais de origem atmosférica. Além disso, diversos setores como da agricultura, aviação, navegação, por exemplo, dependem diariamente das condições climáticas para exercerem suas atividades (INMET, 2014).

2.2 Previsão do Tempo

A previsão do tempo é uma das aplicações da meteorologia para prever o estado da atmosfera em um tempo futuro e em um determinado local. Primeiro é preciso definir o que é o “tempo” na meteorologia: tempo é o estado da atmosfera em determinado instante e lugar. O tempo, portanto, é uma junção de diversos fatores como as condições do ar (umidade, temperatura, pressão), os ventos, a precipitação e as nuvens. As previsões meteorológicas são feitas através da coleta

de dados sobre o estado atual da atmosfera terrestre, e com a compreensão científica dos processos atmosféricos para projetar como o tempo irá evoluir. A alta tecnologia garante mais eficácia nos resultados da previsão do tempo. Com o uso de satélites, computadores e programas para coleta e cálculo de dados atmosféricos é possível ter maior precisão ao estabelecer as condições do tempo. Os institutos de previsão do tempo são responsáveis por coletar dados, como precipitação, ventos, umidade relativa do ar e pressão de uma determinada região. No Brasil, os principais são o Instituto Nacional de Meteorologia (INMET) e o Centro de Previsão de Tempo e Estudos Climáticos (INMET, 2014).

Há uma grande variedade de finalidades para a previsão do tempo. Os avisos de tempo severo são importantes para preservar a vida humana e a economia. As previsões baseadas na temperatura e precipitação são importantes na agricultura. O cotidiano das pessoas pode ser alterado conforme a previsão do tempo.

2.3 Descrição dos Materiais Utilizados

2.3.1 Microcontrolador

Neste projeto foi escolhido como componente principal, a placa de microcontrolador Arduino Uno, que utiliza o microcontrolador ATmega238. Serão utilizadas duas dessas placas, uma será instalada na estação transmissora das informações meteorológicas, com objetivo de processar e enviar os dados para outra placa de Arduino, que será instalado como receptor, servindo como estação de monitoramento. O Arduino Uno possui conexão através de uma porta serial USB, 14 entradas/saídas digitais, 6 entradas analógicas, um conector DC Jack de alimentação e um botão de reset (ARDUINO.CC). Segue a descrição detalhada do *hardware*:

- Microcontrolador: ATmega328;
- Tensão de operação: 5V;

- Tensão de entrada (recomendada): 7-12V;
- Tensão de entrada (limites): 6-20V;
- Pinos de entrada/saída digital: 14;
- Pinos de entrada analógico: 6;
- Corrente contínua por pino entrada/saída: 40 mA;
- Corrente contínua para o pino de 3.3V: 50 mA;
- Memória *Flash*: 32 KB (ATmega328);
- SRAM: 2 KB (ATmega328);
- EEPROM: 1 KB (ATmega328);
- Velocidade do *Clock*: 16 MHz.

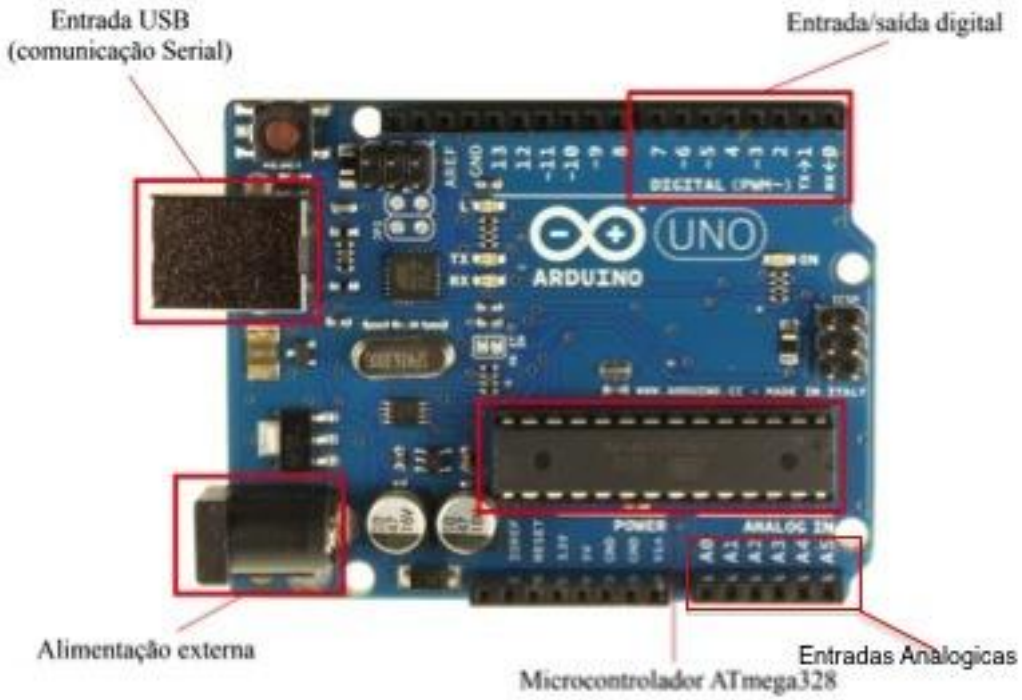

Figura **2-1 – Arduino UNO**

(Fonte: Adaptado de: (ARDUINO.CC))

O microcontrolador ATmega328 possui um conversor A/D (analógico/digital) de 10 bits de resolução para os pinos de entrada analógica, ou seja, é criado para o sinal analógico de entrada, uma representação digital de até 1024 valores diferentes (ARDUINO.CC).

O Arduino Uno possui um regulador de tensão de 5V, o que lhe permite receber como recomendado, uma alimentação variando de 7 a 12 Volts, pois essa tensão de entrada será convertida em uma tensão constante de 5 Volts. A tensão de entrada utilizada no projeto será de 9V (dentro dos limites recomendados 7-12V) e dos 14 pinos digitais de entrada/saída serão utilizados um para estação transmissora e sete para a estação receptora. A placa microcontroladora Arduino Uno possui características importantes para que se alcance o objetivo do projeto, pois além de ter um custo baixo, é *open source,* tornando possível uma grande customização para que se atenda a todos os pré-requisitos do projeto (ARDUINO.CC).

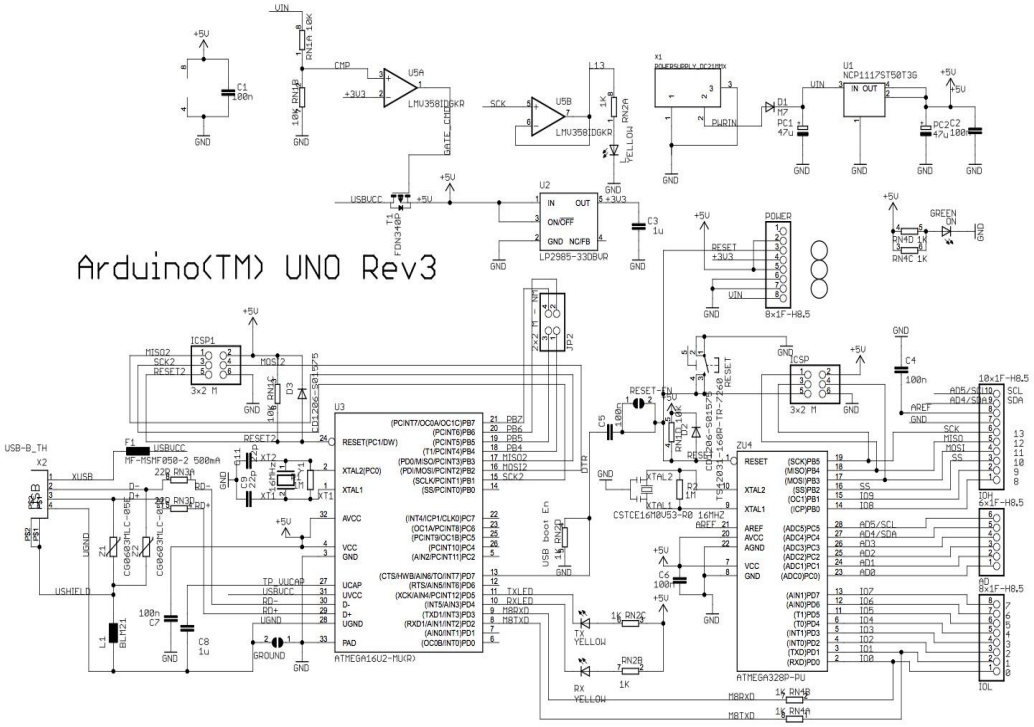

Figura **2-2 – Esquema elétrico Arduino UNO**

(Fonte: (ARDUINO.CC))

2.3.2 IDE do Arduino

O Arduino IDE (*Integrated Development Environment*) é o *software* que nos permite criar os sketches para o Arduino. É a *interface* onde é possível

programar, fazer *debug* do código e o *upload*, para gravar o código no microcontrolador (MCROBERTS, 2011).

É utilizada a linguagem de programação *Wiring*, que é uma linguagem de programação baseada em C e C++. O *software* disponibiliza bibliotecas que facilitam a programação, também é possível adicionar e criar novas bibliotecas. A Figura 2-3 mostra a *interface* do IDE (MONK, 2013).

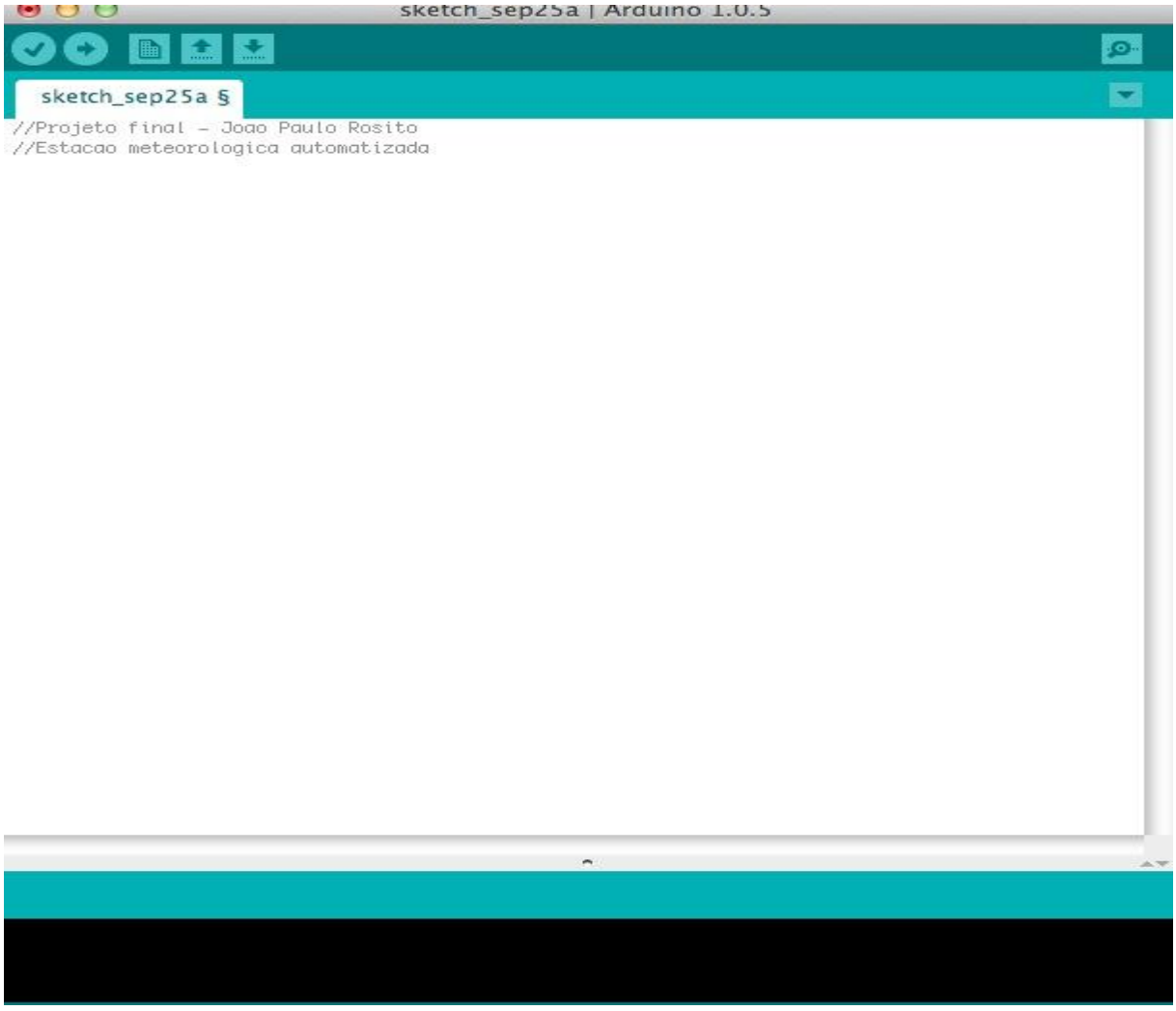

Figura **2-3 – IDE do Arduino**
(fonte: Autor)

2.3.3 Sensor de temperatura

O nível de temperatura está diretamente relacionada com a agitação dos átomos (movimentação dos átomos), sendo que quanto maior a agitação dos átomos, maior a temperatura e quanto menor é essa agitação, menor será a temperatura (ALVARENGA & MAXIMO, 2009).

Para realizar a medição de temperatura será utilizado o sensor LM35, fabricado pela National Semicondutor, pois é um sensor de alta precisão e baixo custo. O A precisão da temperatura pode variar de ¼ ºC a ¾ ºC para mais ou para menos, dentro de uma faixa de temperatura de -55ºC à 150ºC, ou seja, atende ao intervalo de temperatura ao qual o protótipo será submetido. O LM35 é um sensor que não precisa ser calibrado, pois apresenta uma tensão de saída linear de 10 mV para cada grau Celsius, quando alimentado por uma tensão de 4-20 Vdc, o que dispensa conversões de escala de Kelvin para Celsius (TEXAS INSTRUMENTS, 2013).

O sensor LM35 é de fácil acesso no mercado e possui diversos tipos de encapsulamento, pois há uma alta gama de aplicações para este integrado. Neste projeto será usado o sensor com encapsulamento TO-92 por ser o mais comum no mercado e por ter a mesma precisão dos demais.

A Figura 2-4 mostra como são feitas as ligações do Arduino com o sensor de temperatura LM35. Olhando da esquerda para direita temos:

- O pino positivo do sensor que será ligado em 5 V;
- O pino de dados analógicos do sensor que será ligado à um pino analógico do Arduino;
- O pino GND do sensor que será ligado no GND do Arduino;

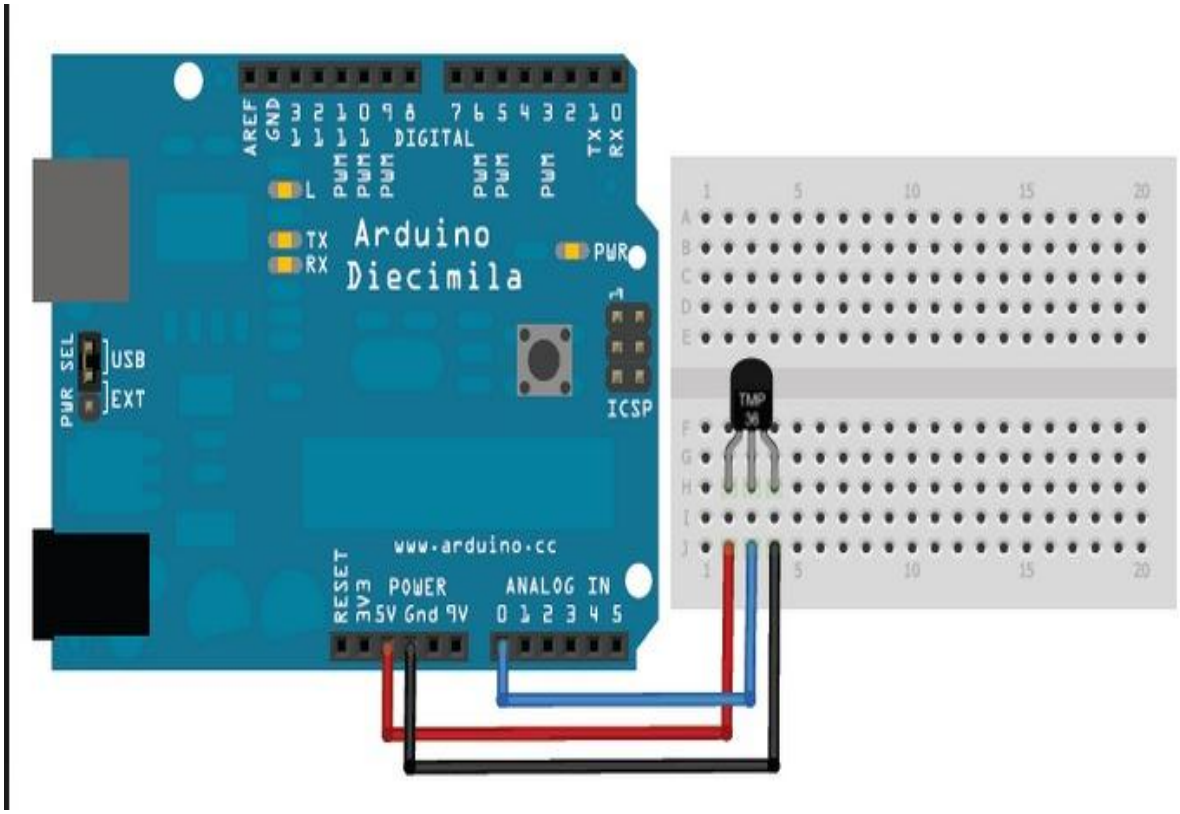

Figura **2-4 – Conexões do sensor LM35 com Arduino**
(fonte: (LADYADA.NET, 2012)

Principais características do sensor LM35:

- É calibrado diretamente em graus Celsius (centígrados);
- Fator de escala linear de + 10,0 mV / °C;
- 0,5 °C de precisão para temperaturas superiores a 25 °C;
- Opera com a faixa de temperatura de -55 ° a 150 ° C;
- Indicado para aplicações remotas;
- Baixo custo;
- Opera de 4 a 20 Volts;
- Necessita de uma corrente de 60 mA;
- Baixo auto-aquecimento, 0,1 °C (no ar);
- Impedância de saída baixa, de 0,1 Ω para 1 mA de carga.

2.3.4 Sensor de pressão

Pressão atmosférica é a pressão que o ar da atmosfera exerce sobre a superfície do planeta e essa pressão pode variar de acordo com a altitude. Quanto maior a altitude, menor a pressão, consequentemente, quanto menor a altitude maior a pressão. A medição da pressão será de fundamental importância para o projeto, tendo em vista que, estamos construindo uma estação meteorológica e que a pressão está diretamente ligada às variações climáticas. Quando temos baixa pressão, significa tipicamente um clima nublado, chuvoso, enquanto em alta pressão a tendência é que se tenha um clima aberto, ensolarado (TOFFOLI, 2014).

No projeto será utilizado o sensor digital de pressão BMP085, desenvolvido pela empresa BOSCH, que possui uma faixa de medição variando de 30.000 – 110.000 Pa (Pascal). O BMP085 possui uma precisão de 2,5 hPa (hectoPascal – 1 hPa equivale a 100 Pa) para mais ou para menos. Além de detectar a pressão atmosférica, o sensor BMP085 também é usado para medir temperatura, que fornece medições variando de 0 – 65ºC, porém, foi escolhido o sensor LM35 como sensor dedicado para esta tarefa, devido ao fato dele ser um sensor mais preciso e não representar um relevante aumento no custo do projeto. A comunicação do sensor com o microprocessador é feita através da *interface* I2C, ou IIC (Inter-Integrated Circuit), o torna a leitura dos dados menos sujeito ao ruído do que um sinal analógico (SENSORTEC, 2009).

Além da medição de pressão atmosférica, outra aplicabilidade para os sensores de pressão está na altimetria, pois como foi visto, a pressão atmosférica está relacionada com a altitude. Será utilizado o sensor BMP085 para realizar também a medida da altitude, sabendo que a pressão ao nível do mar é de 101.325 Pa e que a pressão varia 0,03 hPa para cada 0,25 m, é possível calcular a altitude (SENSORTEC, 2009).

A Figura 2-5 mostra como são feitas as conexões do sensor de pressão BMP085 com o Arduino. Da esquerda para direita temos:

- O pino GND do sensor ligado ao GND do Arduino;

- O pino SCL do sensor ligado à um pino analógico do Arduino;
- O pino SDA do sensor ligado à um pino analógico do Arduino;
- O pino VCC (positivo) do sensor ligado em 3,3V;

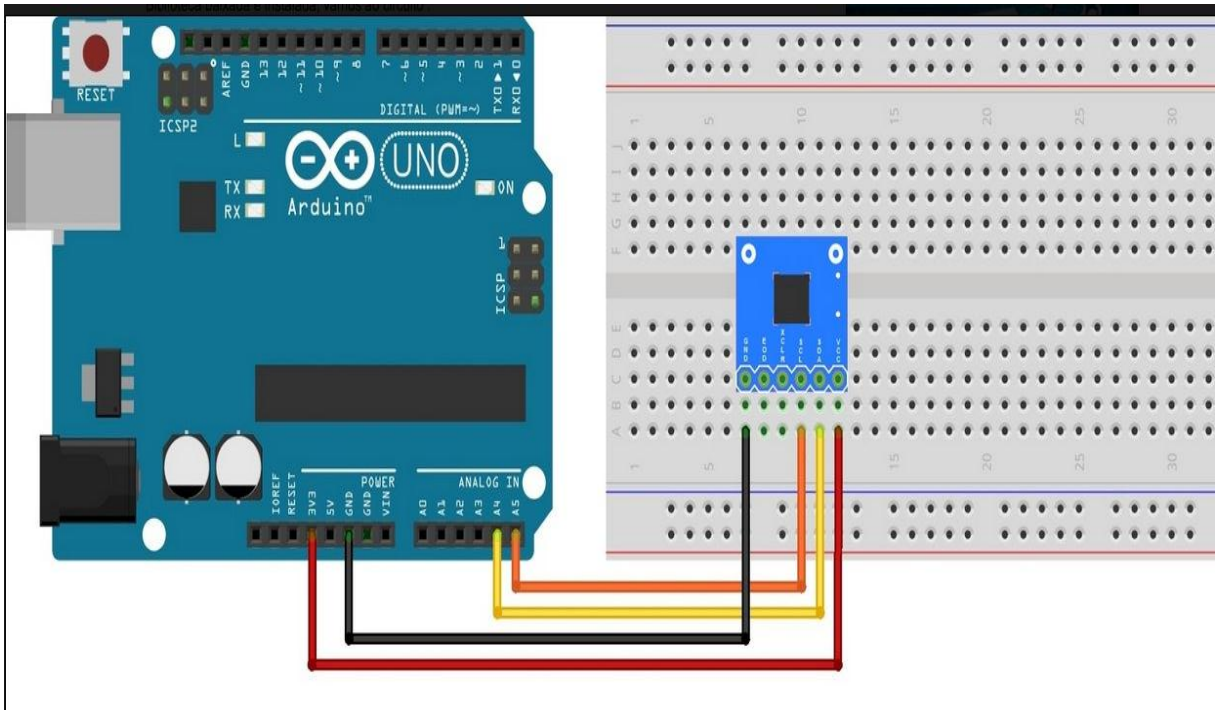

Figura **2-5 – Conexões do sensor BMP085 de pressão com o Arduino (fonte: (ARDUINO E CIA))**

2.3.5 Sensor de umidade

Umidade relativa é um termo bastante comum nos informativos meteorológicos, porque a umidade é de fundamental importância para que se realize previsões climáticas. Para entender o que é umidade relativa é necessário entender o que é umidade absoluta. Umidade absoluta é a relação entre a massa do vapor de água e a massa de ar seco, ou seja, é a quantidade de água existente em um determinado volume de ar. Umidade relativa é a relação entre a umidade absoluta e a quantidade máxima de água que poderia haver neste volume de ar, na mesma

temperatura em que foi calculada a umidade absoluta, comumente expressa em porcentagem (WETHERLY, 2008).

Para que possamos obter a umidade relativa do ar será usado o sensor DHT11, que é um sensor de umidade e temperatura. O DHT11 utiliza um sensor capacitivo de umidade e um termistor para medir a temperatura do ar, ambos conectados a um controlador de 8 bits. Apesar de o sensor DHT11 também medir temperatura, essa tarefa será atribuída ao sensor LM35 por ter uma precisão maior, melhor tempo de resposta e uma faixa de medição menos limitada. O DHT11 permite medir temperaturas de 0 a 50 ºC com precisão de 2 graus para mais ou para menos, e umidade nas faixas de 20 a 90%, com precisão de 5% para mais ou para menos. A Figura 2-6 ilustra os pinos e como serão feitas as conexões do sensor com o Arduino. Seguem descrições detalhadas do sensor e conexões:

Descrições do sensor DHT11

- Alimentação de 3 V -5 V;
- Faixa de medição de umidade: 20 a 90% UR (Umidade Relativa);
- Faixa de medição de temperatura: 0 a 50 ºC;
- Precisão de umidade +- 5%;
- Precisão de temperatura +- 2 ºC;
- Tempo de resposta do sensor maior que 2 segundos e menor que 5 segundos;

Conexões do sensor com o Arduino

- Pino VCC do sensor será conectado em uma alimentação de 3,3 V;
- Pino Data do sensor será conectado à um pino analógico do Arduino;
- Pino GND do sensor será conectado ao GND do Arduino;

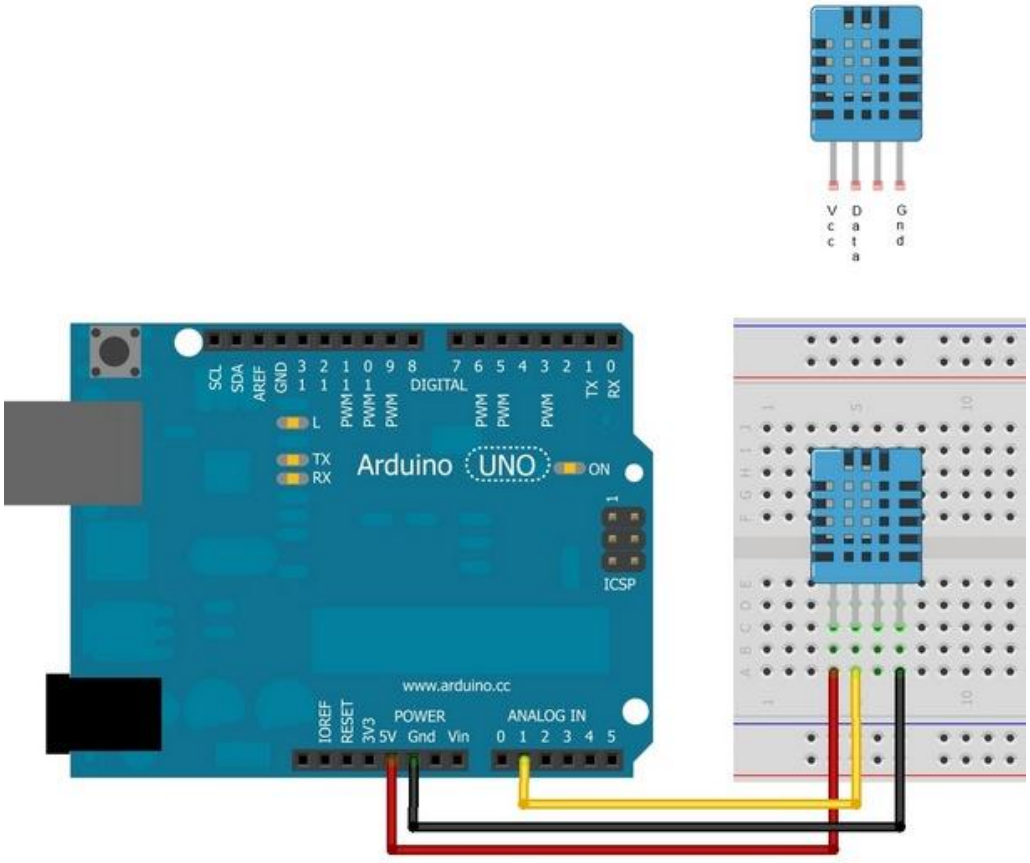

Figura **2-6 – Conexões do sensor DHT11 de umidade com o Arduino (fonte: (ARDUINO E CIA))**

2.3.6 Visor LCD

No projeto será usado um visor LCD 16x2 (16 colunas por 2 linhas) da marca HITACHI modelo HD44780U que possui letras pretas e fundo verde para exibição dos dados meteorológicos na estação base de coleta das informações, a fim de tornar possível o monitoramento local. Este *display* foi escolhido pois é compatível com a biblioteca LiquidCrystal do Arduino, o que facilita bastante a programação, além de ser um *display* de baixo custo e facilmente encontrado no mercado. O *display* possui *interface* paralela, ou seja, o microcontrolador precisa controlar os pinos de *interface* em vez de ter que controlar o visor. O código desta *interface* pode ser encontrado facilmente na internet e de forma gratuita. Os pinos desta *interface* estão ilustrados conforme a Figura 2-7 e descritos a seguir: (ARDUINO.CC)

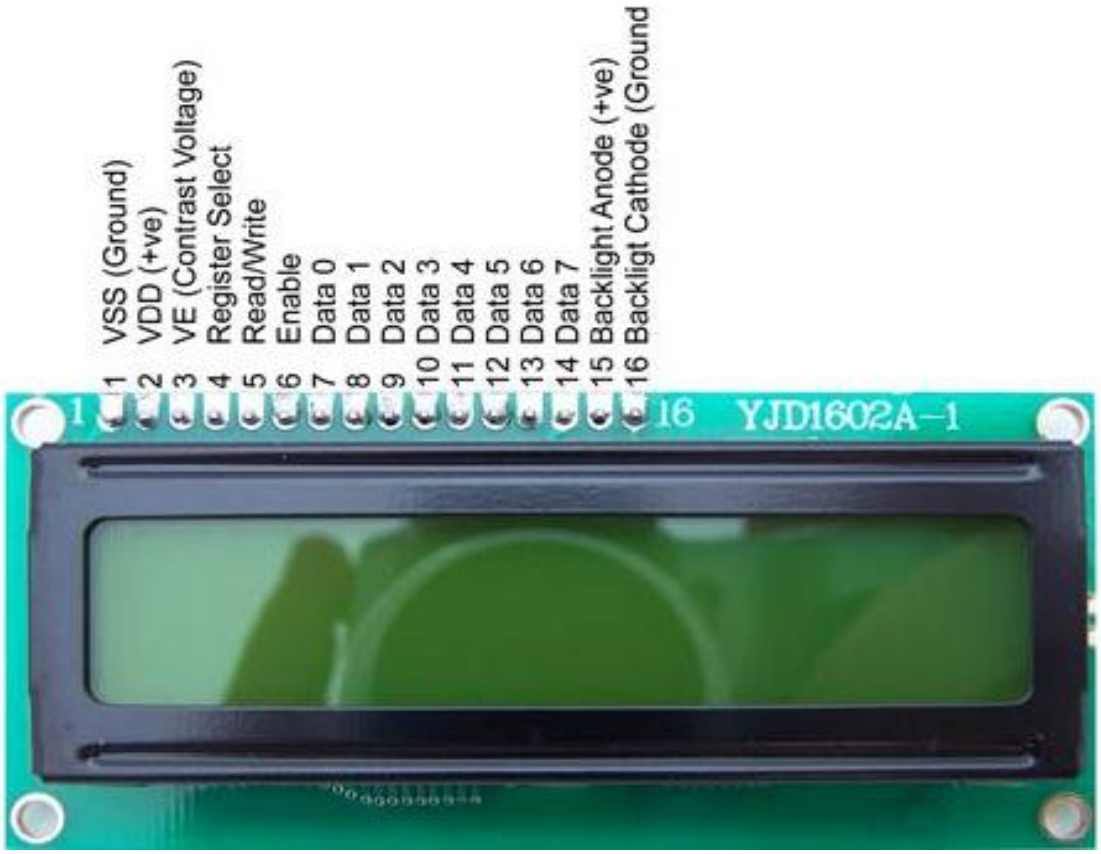

Figura **2-7 – *Display* LCD 16x2**
(fonte: (PROTOSTACK, 2014))

- RS (register select) Pino que controla onde na memória do LCD os dados serão gravados;
- R/W (Read/Write) Pino que seleciona o modo leitura ou o modo escrita;
- E (Enable) Pino que permite que os registros sejam gravados;
- D0 – D7 (8 pinos de dados) O estado desses pinos (high ou low) são os bits que você está escrevendo ou os valores que você está lendo;
- VE pino de contraste do *display*;
- VDD pino do positivo onde será ligado uma alimentação de 5 V;
- VSS (GND) pino onde será ligado o terra;
- LED+/LED- pinos que podem ser usados para ligar e desligar a luz de fundo do *display*;

As conexões do *display* LCD com o microcontrolador Arduino serão feitas conforme Figura 2-8 e descritas de forma detalhada a seguir:

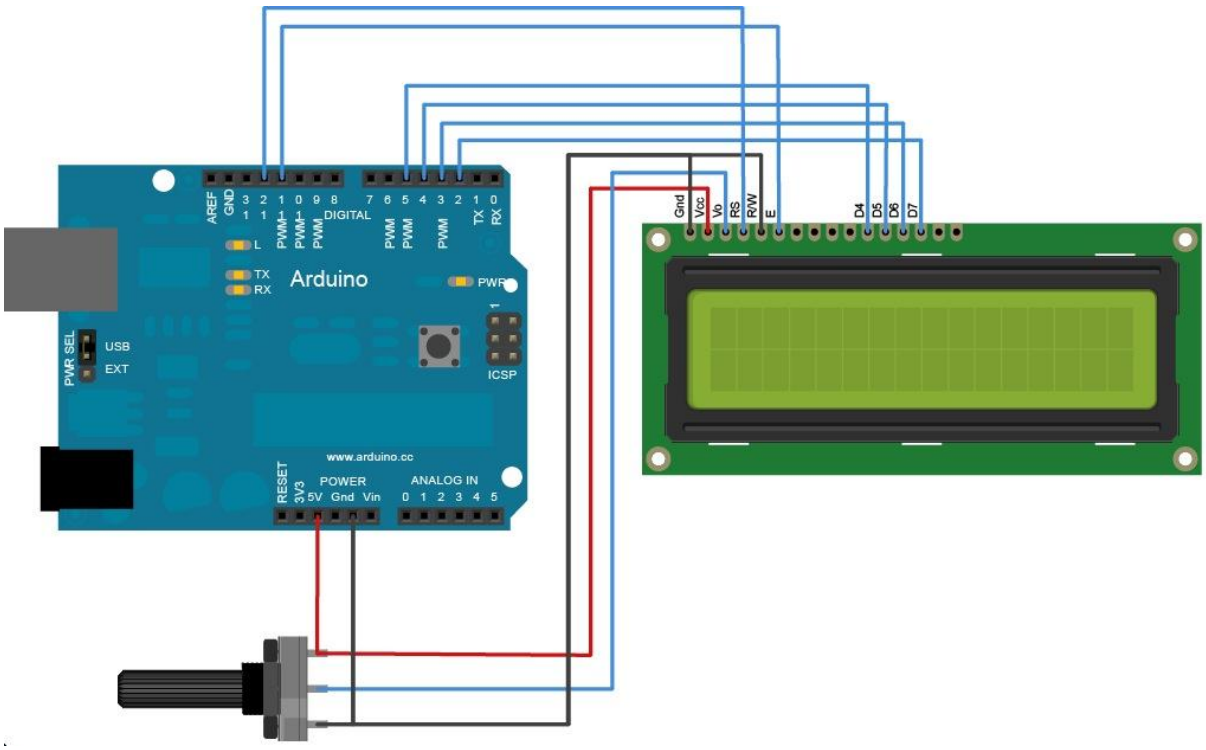

Figura **2-8 – Conexões do *display* LCD com o Arduino**
(Fonte: (ARDUINO.CC))

- Pino RS do LCD ligado ao pino digital 12 do Arduino;
- Pino E do LCD ligado ao pino digital 11 do Arduino;
- Pino D4 do LCD ligado ao pino digital 5 do Arduino;
- Pino D5 do LCD ligado ao pino digital 4 do Arduino;
- Pino D6 do LCD ligado ao pino digital 3 do Arduino;
- Pino D7 do LCD ligado ao pino digital 2 do Arduino;
- Pino VDD do LCD ligado em uma alimentação de 5 V;
- Pino VSS do LCD ligado ao pino GND do Arduino;
- Pino VE do LCD ligado ao pino GND do Arduino;

2.3.7 Módulo de Radiofrequência

Para o desenvolvimento do projeto, fez-se necessária a transmissão de dados entre as duas estações em um meio que não utilize contato físico (fios), para que não tenhamos problemas com infraestrutura, para isso pode-se utilizar várias técnicas como infravermelho ou ondas de rádio. Como meio de comunicação dos dados meteorológicos entre as estações transmissora e receptora, será usada a transmissão por ondas de rádio. No mercado existem módulos de rádio prontos, que possuem uma faixa de frequência permitida no Brasil pela ANATEL, trabalhando em frequências como por exemplo: 315 MHz e 433 MHz. dentre outras. No projeto será escolhido o módulo *link* de radiofrequência de 433 MHz. Este é um módulo de radiofrequência simples, onde o transmissor envia dados em série para o receptor utilizando o método de transmissão simplex, ou seja, transmissão unidirecional (THOMANZINI & ALBUQUERQUE, 2008).

2.3.7.1 Módulo Transmissor

O módulo transmissor será instalado na estação de coleta das informações e enviará para estação base de monitoramento. Os pinos do módulo transmissor da esquerda para direita são: Data (pino de dados), VCC (pino de alimentação) e GND (ground), conforme ilustrado na Figura 2-9. Seguem detalhadas as especificações deste módulo:

- Modelo MX-FX03V;
- Alcance de 20 – 200 metros (dependendo da voltagem, no projeto usaremos 5V);
- Opera em tensões de 3,5 – 12 V;
- Modulação ASK;
- Taxa de transferência de 4 Kb/s;
- Frequência de transmissão de 433 MHz;

Figura **2-9 – Módulo Transmissor de Radiofrequência**
(fonte: http://www.filipeflop.com)

A Figura 2-10 ilustra como serão feitas as conexões do módulo transmissor com o Arduino:

- Pino Data do módulo transmissor será conectado à um pino digital do Arduino;
- Pino VCC do módulo transmissor será alimentado em 5 V;
- Pino GND do módulo será conectado ao GND do Arduino;

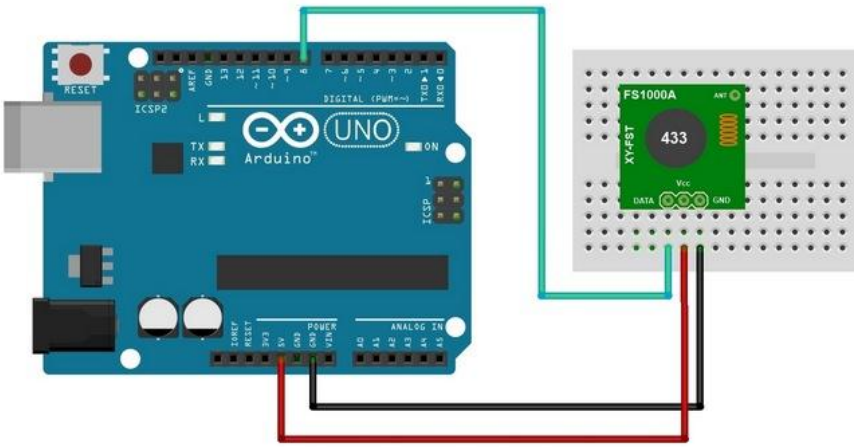

Figura **2-10 – Conexão do módulo transmissor com Arduino (fonte: (FILIPEFLOP, 2014))**

2.3.7.2 Módulo Receptor

O módulo receptor será instalado na estação de monitoramento dos dados meteorológicos, e será o responsável por receber o sinal enviado. Os pinos do módulo receptor da esquerda para direita são: VCC (pino de alimentação), Data (pino de dados), Data (pino de dados) e GND (groud), conforme ilustrado na Figura 2-11. Seguem detalhadas as especificações deste módulo:

- Modelo MX-05V;
- Tensão de operação de 5 V;

- Frequência de recepção de 433 MHz;
- Corrente de operação de 4 mA;

Figura **2-11 – Módulo Receptor de Radiofrequência (fonte (FILIPEFLOP, 2014))**

A Figura 2-12 ilustra como serão feitas as conexões do módulo receptor com o Arduino:

- Pino GND do módulo receptor ligado ao GND do Arduino;
- Pino Data do módulo receptor ligado à um pino digital do Arduino;
- Pino VCC do módulo receptor ligado em uma alimentação de 5 V;

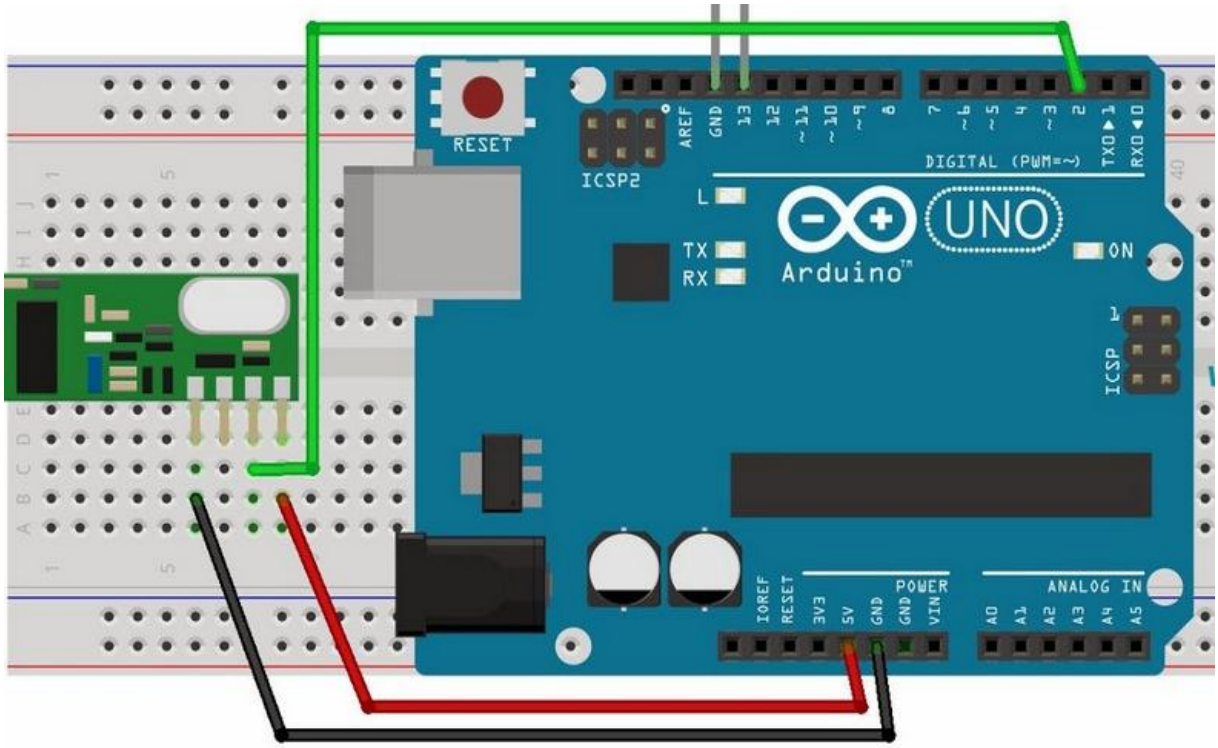

Figura **2-12 – Conexões do módulo receptor com o Arduino (fonte: (GATEWAYIT, 2014))**

2.3.8 *Ethernet Shield*

A utilização do *Ethernet Shield* tornará possível fazer uma conexão do Arduino com um roteador e transmitir os dados via cabo *Ethernet*. Ao fazê-lo, será possível ler as informações dos sensores de qualquer ponto da rede através de uma *interface web*. Essa será a solução adotada para monitoramento das informações meteorológicas na estação receptora. Essa placa tem a vantagem de ser de fácil integração com o micrcontrolador Arduino Uno, pois tem o formato de um *shield* (formato empilhável), onde é encaixada sobre o microcontrolador Arduino, conforme ilustrado na Figura 2-13.

O *Shield* possui um controlador W5100 que permite criar conexão do Arduino com Internet e/ou dispositivos em rede local. Esse *Shield* é equipado com um soquete para cartão micro-SD, que pode ser usado para armazenamento de dados. A comunicação com o Arduino é feita através da *interface* SPI (Serial

Peripheral *Interface*), que é um protocolo utilizado para comunicação entre microcontroladores ou de um microcontrolador para um ou mais periféricos.

Figura **2-13 – Conexão do *Ethernet Shield* com o Arduino (fonte: (WAIHUG.NET))**

Seguem as descrições detalhadas do *Ethernet Shield*:

- Conector RJ45;
- Soquete para cartão micro-SD;
- Controlador *Ethernet* W5100 com buffer interno de 16 k;
- Velocidade de conexão de 10/100 Mb;
- Tensão de operação de 3,3 – 5 V;
- Suporta até 8 conexões TCP/UDP simultâneas;
- Compatível com a biblioteca *Ethernet* do Arduino;
- Compatível com Arduino Uno (usado no projeto);

2.4 Sistema de Controle

Um sistema de controle é formado pela interconexão de componentes afim de produzir uma resposta desejada. O sistema a ser controlado é chamado de processo, o sinal de entrada que será aplicado no processo é chamado de variável manipulada e o sinal de saída do processo é chamado de variável controlada. No projeto as variáveis manipuladas serão os dados meteorológicos coletados pelos sensores instalados na estação transmissora, o processo será o sistema da estação meteorológica que coletará e manipulará os dados e o sinal de saída (variável controlada) serão os dados processados e exibidos para o usuário em um *display* LCD e *interface web*. Sistema de controle de malha aberta é um sistema onde o sinal de saída não exerce nenhuma ação de controle, ou seja, o valor dele não será realimentado pelo sistema para ser comparado com o sinal de entrada a fim de resultar em um controle (DORF & BISHOP, 2013).

Este projeto será um sistema de controle de malha aberta, pois teremos os sinais de entrada (dados meteorológicos) o processamento dessas informações que será o controle feito e uma saída que será a amostragem das informações meteorológicas em um *display* LCD e em uma *interface web*, conforme ilustrado na Figura 2-14.

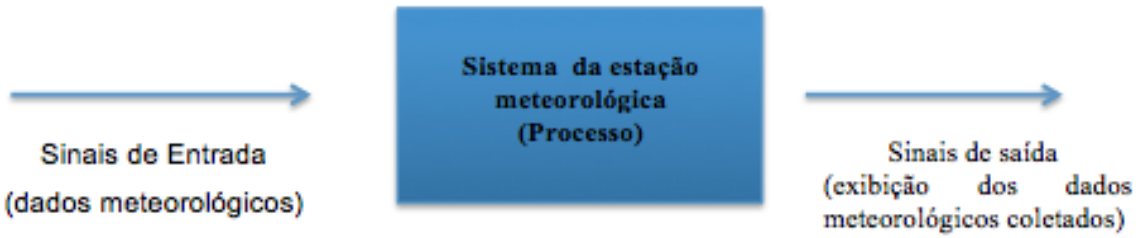

Figura **2-14 - Esquemático do sistema de controle de malha aberta**
(Fonte: Autor)

CAPÍTULO 3 DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo tem por objetivo mostrar a solução proposta, todas as etapas do desenvolvimento, ilustrando tudo que for feito com base na aplicação dos conceitos teóricos adquiridos e apresentados no capítulo 2.

3.1 Apresentação Geral do Desenvolvimento do Protótipo

Para solucionar o problema apresentado (criação de uma estação meteorológica automatizada de baixo custo em regiões sem infraestrutura adequada e que tenham riscos de deslizamentos de terra e enchentes), propõe-se a criação do seguinte protótipo: Duas estações (transmissora e receptora) com comunicação por radiofrequência, onde a transmissora coletará os dados meteorológicos e enviará para receptora que por sua vez disponibilizará para o usuário através de um *display* LCD e via *interface web* em rede local para que seja feito o monitoramento. As etapas do desenvolvimento do projeto estão ilustradas na Figura 3-1.

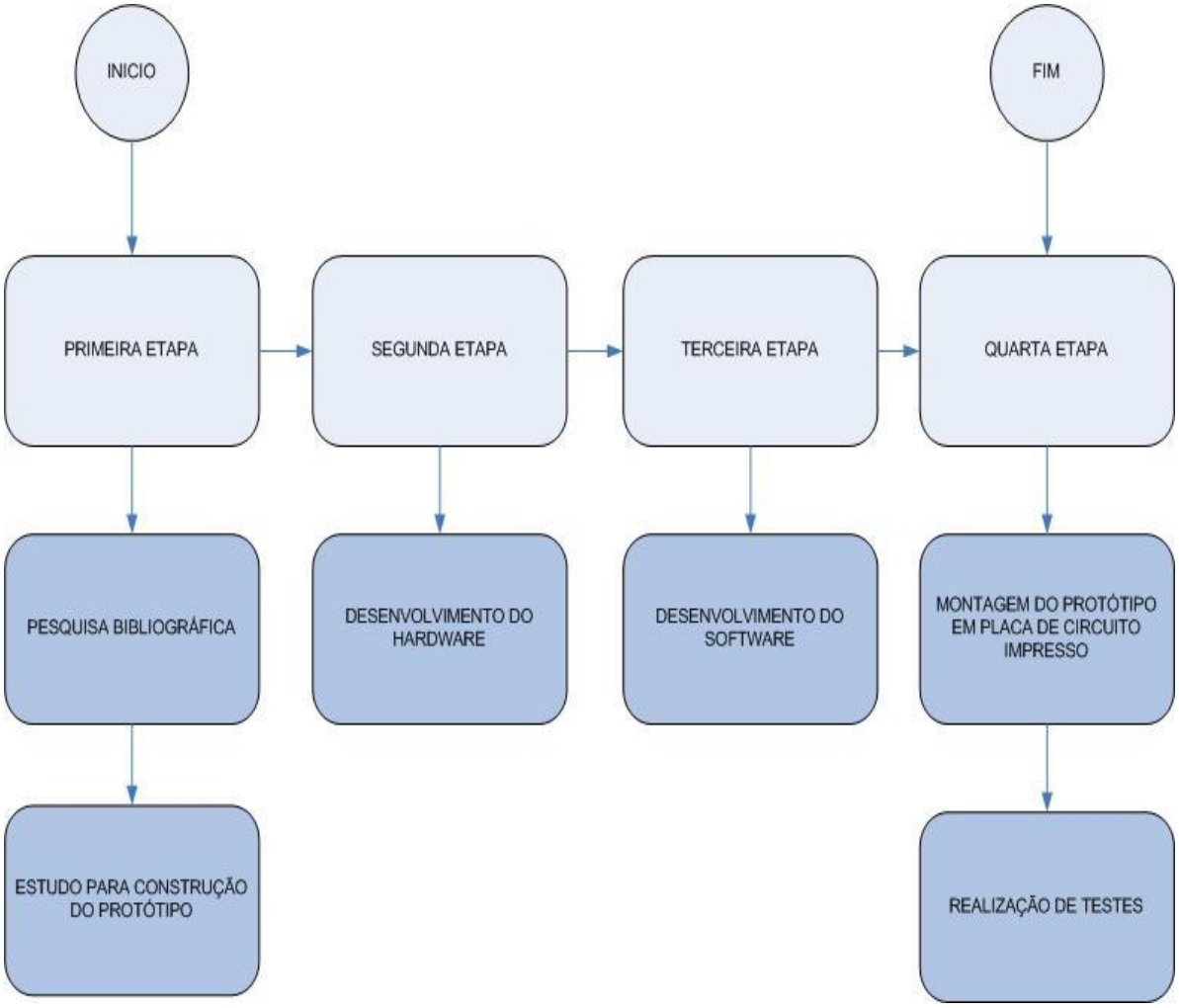

Figura 3-1 – Etapas do projeto
(Fonte: Autor)

3.2 Construção da Estação Transmissora

A estação transmissora será a responsável por coletar os dados meteorológicos, processá-los, e transmiti-los via radiofrequência para a estação base. A Figura 3-2 representa um esquemático desta estação. Para construção da estação transmissora, foram utilizados sensores de temperatura, umidade, pressão e um módulo de radiofrequência para transmissão dos dados.

Figura **3-2 – Esquemático da estação transmissora**
(Fonte: Autor)

3.2.1 Sensor de Temperatura

Nesta etapa foi instalado o sensor de temperatura LM35 ao Arduino, para que possamos obter a temperatura ambiente. O sensor LM35 possui 3 pinos, como dito e ilustrado no capítulo 2 (VCC, pino dados e GND). O pino VCC foi alimentado em 5V (*jumper* vermelho), o pino de dados foi conectado ao pino analógico A1 do Arduino (*jumper* branco) e o pino GND foi ligado ao GND do Arduino (*jumper* azul), conforme ilustrado na Figura 3-3.

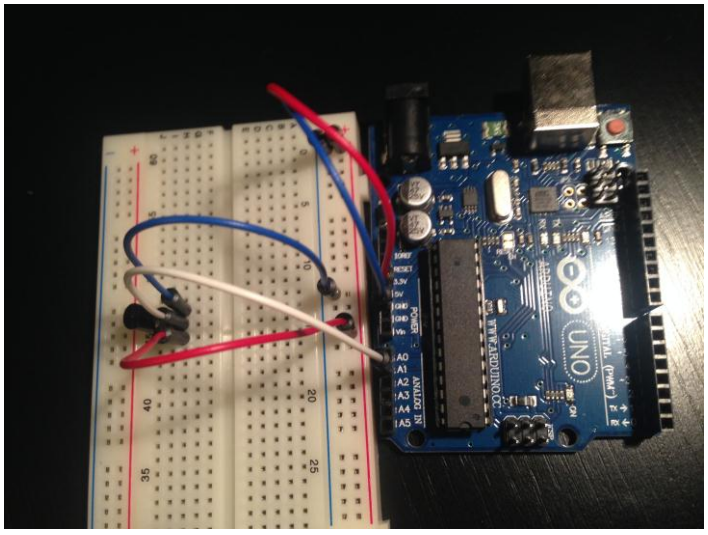

Figura **3-3 – Montagem do sensor LM35**
(Fonte: Autor)

3.2.2 Sensor de Umidade

Para coletar a umidade relativa do ar foi instalado ao protótipo o sensor de umidade DHT11. Este sensor possui 4 pinos porém usamos 3 (VCC, pino de dados e GND). O pino VCC será alimentado em 3,3 V (*jumper* vermelho), o pino de dados será conectado ao pino analógico A2 do Arduino (*jumper* branco) e o GND será ligado ao GND do Arduino (*jumper* azul), conforme ilustrado na Figura 3-4.

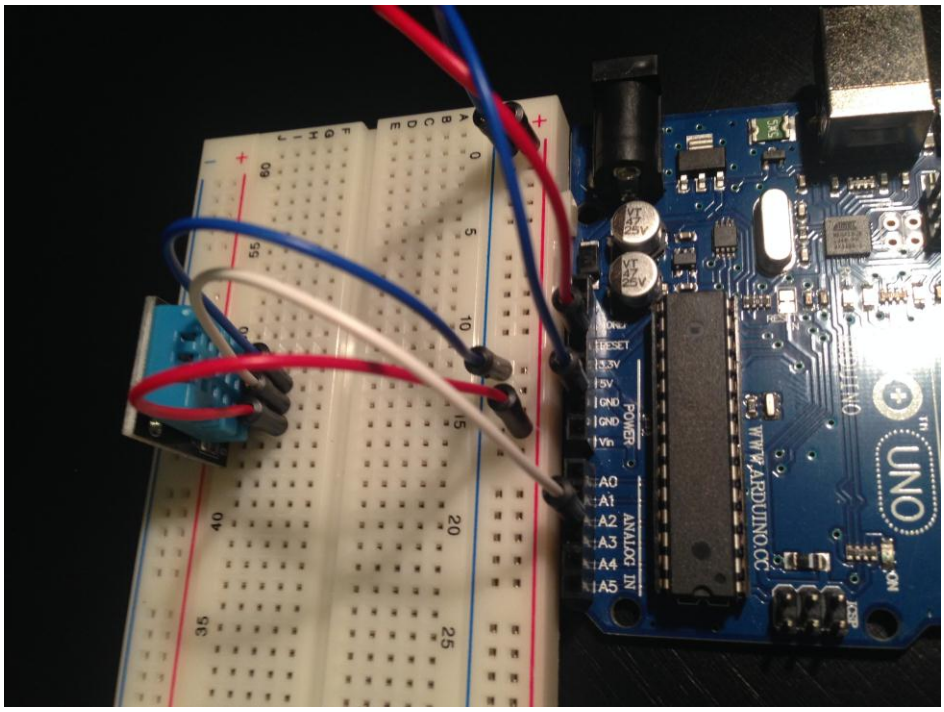

Figura **3-4 – Montagem do Sensor DHT11 de umidade**
(Fonte: Autor)

3.2.3 Sensor de Pressão

Foi utilizado o sensor BMP085 para obtenção da pressão atmosférica e altitude. No projeto usaremos 4 dos 6 pinos que o sensor BMP085 tem. O pino GND

do sensor será ligado ao GND do Arduino (*jumper* azul), o pino SCL do sensor será conectado ao pino analógico A5 do Arduino (*jumper* verde), o pino SDA do sensor será conectado no pino analógico A4 do Arduino (*jumper* preto) e o pino VCC será alimentado em 3,3 V (*jumper* vermelho), conforme ilustra a Figura 3-5.

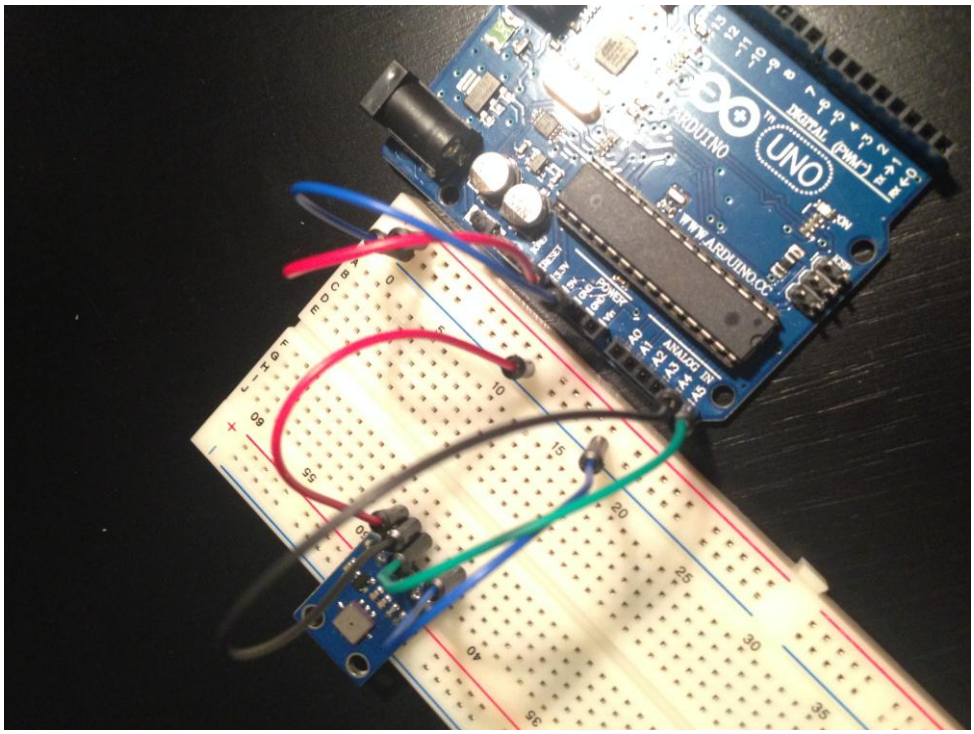

Figura **3-5 - Montagem do sensor BMP085 de pressão**
(Fonte: Autor)

3.2.4 Módulo Transmissor de radiofrequência

Para envio dos dados para estação receptora foi usado um módulo *link* de radiofrequência 433 MHz. O módulo transmissor tem 3 pinos, um pino de dados que será ligado na porta digital 8 do Arduino (*jumper* branco), um pino VCC que será alimentado em 5 V (*jumper* vermelho) e um pino GND que será ligado ao GND do Arduino (*jumper* azul), conforme ilustra a Figura 3-6. Para construção do protótipo foi usado um módulo de 433 MHz com uma alimentação de 5 V onde conseguimos um

alcance de 50 metros, mas para uma instalação em condições reais o mais recomendado seria usar um transmissor de rádio com frequências maiores, de 2,4 GHz ou 5 GHz, para que se tenha maiores alcances de sinal, na faixa de um ou mais quilômetros.

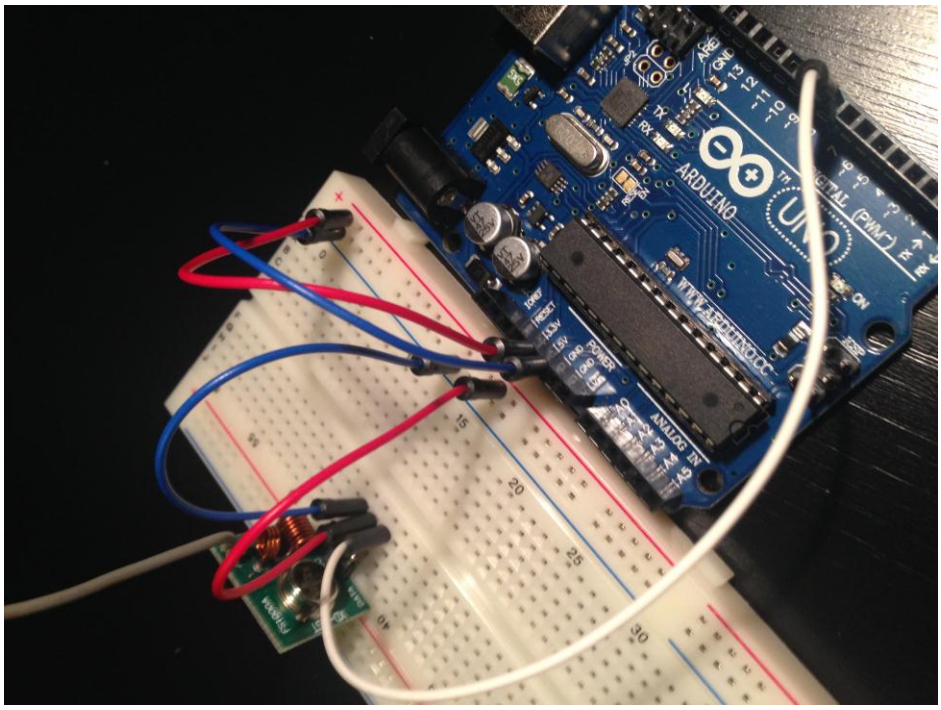

Figura **3-6 - Montagem do módulo transmissor de RF**
(Fonte: Autor)

3.3 Construção da Estação Receptora

A estação receptora será a responsável por receber os dados meteorológicos transmitidos por radiofrequência e mostrar para o usuário em um *display* LCD e *interface web* via rede local instalada. A Figura 3-7 representa um esquema desta estação.

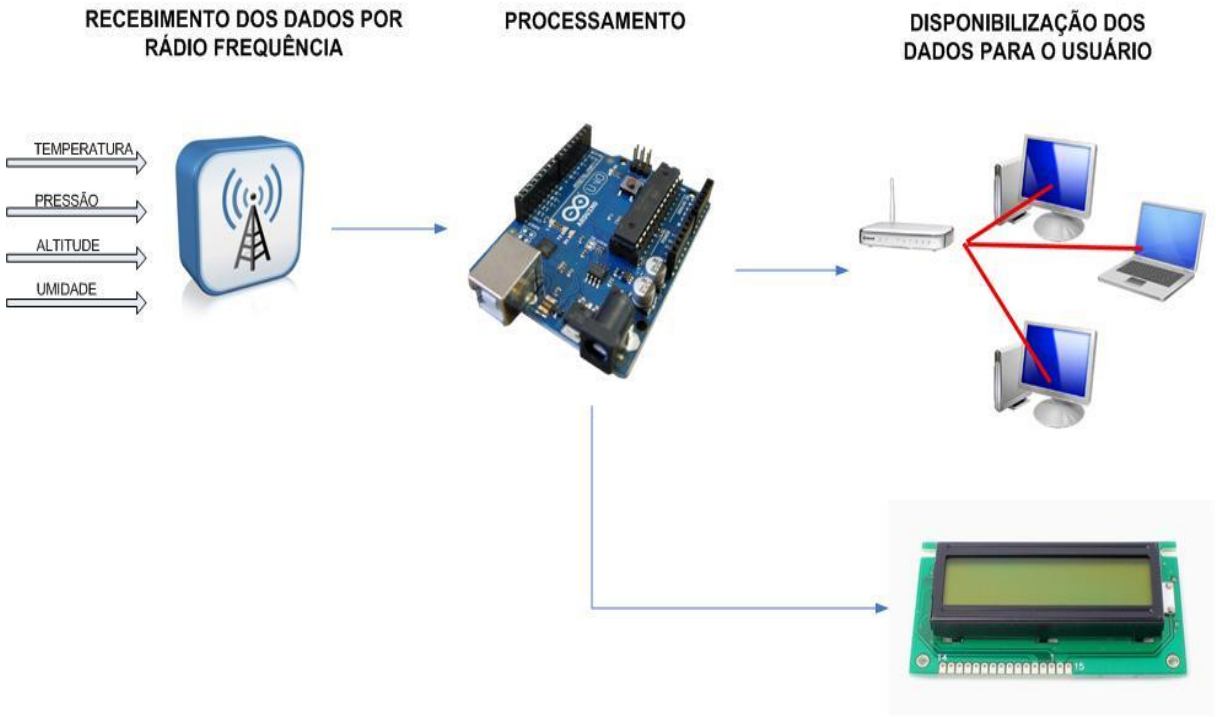

Figura 3-7 – Esquema da Estação Receptora
(Fonte: Autor)

3.3.1 Módulo Receptor de Radiofrequência

Para receber as informações enviadas pela estação transmissora, foi usado um módulo receptor de radiofrequência de 433 MHz. Este módulo possui 4 pinos, mas no projeto foram usados 3, o pino VCC de alimentação do módulo receptor que foi ligado em 5 V (*jumper* vermelho), um pino de saída para os dados que foi conectado à porta digital 8 do Arduino (*jumper* amarelo) e um pino GND que foi conectado ao GND do Arduino (*jumper* azul), conforme ilustrado na Figura 3-8

.

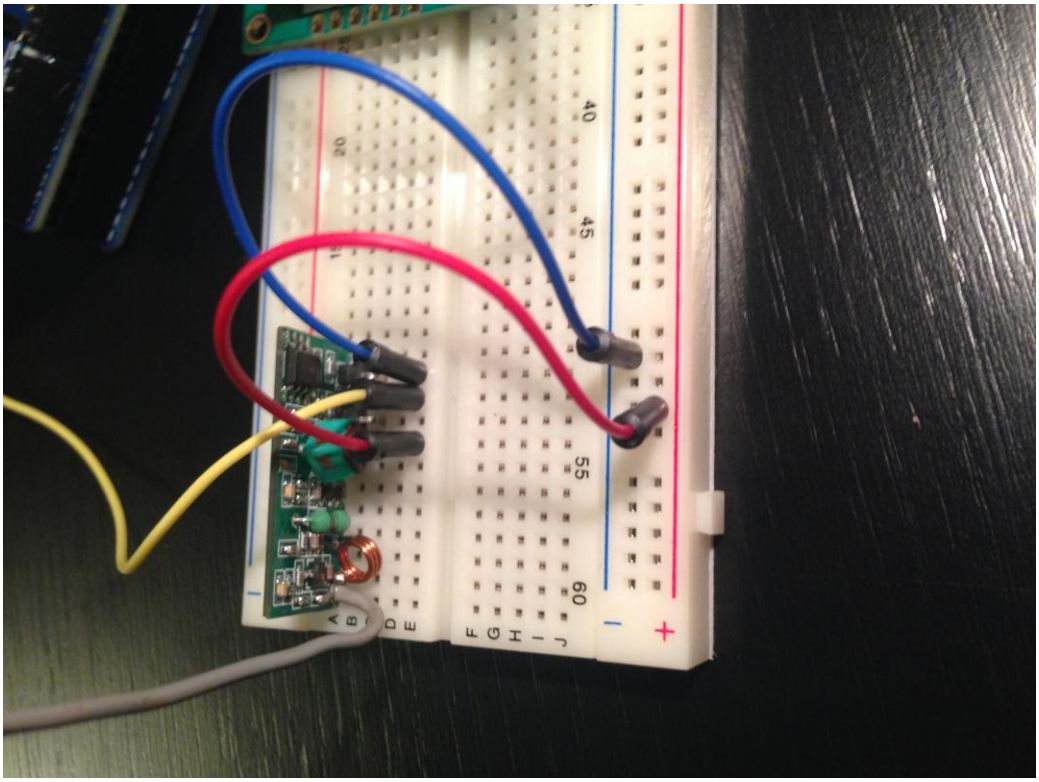

Figura 3-8 – Montagem do Módulo Receptor de RF
(Fonte: Autor)

3.3.2 *Display* LCD

O *display* LCD será instalado na estação base para monitoramento local. No projeto foi usado um *display* LCD 16x2 que possui 16 pinos. As conexões do *display* com o Arduino estão descritas no *Quadro* a seguir:

***Quadro* 1- Conexões do *display* LCD com o Arduino**

Display LCD	Arduino
VSS	GND
VDD	5V
VE	GND
RS	Digital 1
RW	GND
Enable	Digital 0
Data 4	Digital 5
Data 5	Digital 4
Data 6	Digital 3
Data 7	Digital 2

(Fonte: Autor)

Este *display* possui 2 linhas e 16 colunas, na primeira tela será mostrado a temperatura na primeira linha e a pressão na segunda, a tela será limpada e após alguns segundos será mostrado a altitude na primeira linha e a umidade na segunda linha.

3.3.3 *Ethernet Shield*

A solução proposta para atender a necessidade de monitoramento das informações recebidas é a utilização do *Ethernet Shield*, que conforme visto no capítulo 2, torna possível a conexão do Arduino a um roteador. É criada uma *interface web* para o monitoramento dos parâmetros meteorológicos pelo usuário em rede local.

O *Shield* foi conectado sobre o Arduino conforme ilustrado na Figura 3-9 e possui um conector RJ45 que foi ligado a um roteador.

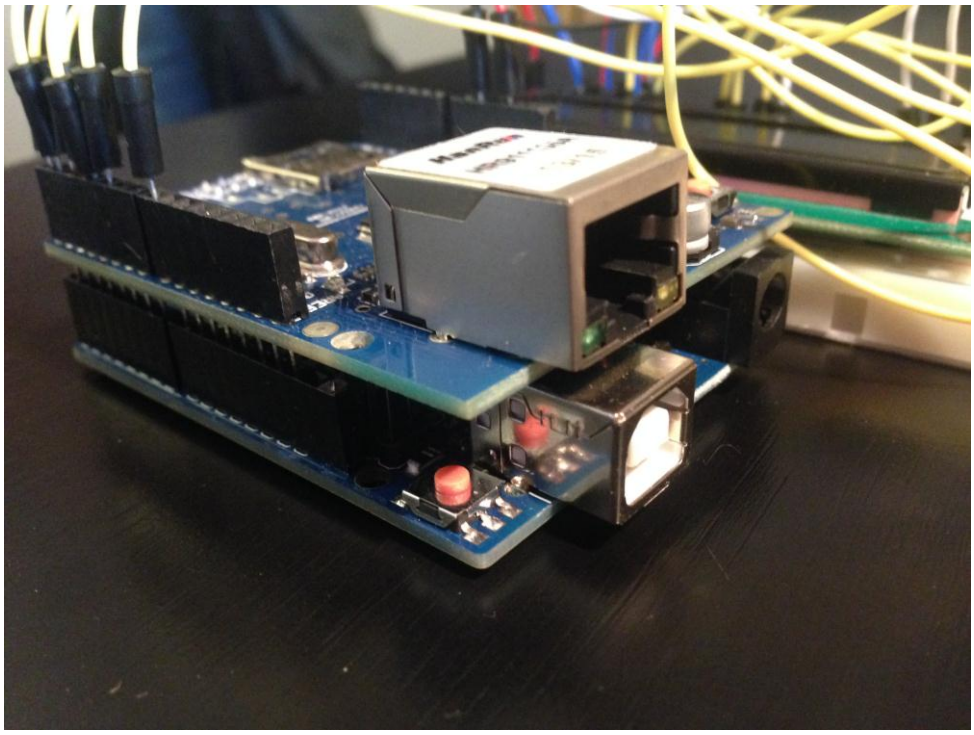

Figura 3-9 – Encaixe do *Ethernet Shield* ao Arduino
(Fonte: Autor)

3.3.4 Rede Local

No projeto foi usado o roteador Dl-624 da marca D-*Link* para receber as informações do Arduino e distribuir para o usuário em rede local, ou seja, todos os usuários da rede, com um limite de quatro acessos simultâneos, serão capazes de inserir o endereço de IP do Arduino em uma página HTML e visualizar por meio de uma *interface web*, as informações meteorológicas para monitoramento. No *Quadro* 3.2 estão descritas as conFigurações do Arduino para acesso à rede.

***Quadro* 2- Configurações do Arduino na rede**

Endereço MAC	0xDE:0xAD:0xBE:0xEF:0xFE:0xED
Endereço de IP	192.168.100.177
Máscara de sub-rede	255.255.255.0
Endereço do Gateway	192.168.100.1

(Fonte: Autor)

3.4 Desenvolvimento do *Software*

Para o desenvolvimento do *software* foi utilizado a plataforma de desenvolvimento do Arduino, que utiliza a linguagem de programação *Wiring* (baseada em C/C). Com a utilização do IDE do Arduino é possível desenvolver o programa e passá-lo para o microcontrolador via USB para que ele seja executado.

A seguir será descrito detalhadamente todas as etapas de construção do *software*, que foi dividida em duas partes: *Software* da estação transmissora e *software* da estação receptora.

3.4.1 Estação Transmissora

No projeto serão necessários a criação de dois Sketches (programas), pois serão usados dois Arduinos com um Sketch em cada. Na construção do Sketch da estação transmissora é feita a leitura dos valores detectados pelos sensores (pressão, umidade e temperatura), cálculo do valor da altitude e envio dos dados via radiofrequência para a estação receptora. Seguem os passos detalhados da construção.

3.4.1.1 Inclusão das Bibliotecas

O *software* de desenvolvimento do Arduino, assim como outros, possibilita que o usuário utilize bibliotecas, que são funções já escritas por outros programadores com o objetivo de facilitar a programação. A este conjunto de funções pré-escritas é dado o nome de biblioteca ou *library*. A instalação normal do IDE do Arduino já é composta por diversas bibliotecas, no entanto é possível importar novas ou até mesmo criar.

Para o desenvolvimento da estação transmissora foi necessário a utilização das seguintes bibliotecas:

- VirtualWire.h - responsável pelo reconhecimento das funções do módulo de radiofrequência;
- Wire.h e Adafruit_BMP085.h - possibilita utilizar as funções do sensor de pressão BMP085;
- DHT.h - possibilita a utilização das funções do sensor de umidade DHT11.

O Código 3-1 representa todas as bibliotecas que foram utilizadas para o desenvolvimento do código da estação transmissora.

```
#include <VirtualWire.h> // Biblioteca do modulo de Radiofrequência
#include <Wire.h> // Biblioteca do barômetro
#include <Adafruit_BMP085.h> // Biblioteca do barômetro
#include "DHT.h" //Biblioteca do sensor de umidade
```

Código 3.1 – Biblioteca da Estação Transmissora

(fonte:Autor)

3.4.1.2 Inclusão dos sensores

O programa é iniciado com a chamada da função *void setup* ( ), essa função é utilizada para iniciar variáveis, pinos e o uso da biblioteca. É uma função

que é executada apenas uma vez, quando o microcontrolador é ligado e todas as vezes que ele for reiniciado.

O reconhecimento dos valores coletados pelos sensores é feito dentro da estrutura *void loop*, que realiza *loops* infinitos ou até que se atinja uma condição. No caso da estação transmissora, esses *loops* serão infinitos. A função dele é ficar sempre coletando as informações dos sensores e enviado para a outra estação. O Código 3-2 representa as funções responsáveis por detectar as informações dos sensores (temperatura, pressão, altitude e umidade) e a atribuição dessas informações às variáveis.

```
void loop() {
  int D = analogRead(LM35); // leitura analógica do sensor LM35
  int T = (D * 500) / 1023; // Conversão da tensão de entrada para temperatura em graus
  int P = (bmp.readPressure()); // leitura da pressão
  int H = dht.readHumidity(); // leitura da umidade
  int A = (bmp.readAltitude()); // leitura da altitude
```

Código 3.2 – Detectando o valor dos sensores

O sensor de temperatura (LM35) possui um sinal de saída de tensão linear, que varia 10 mV para cada grau de temperatura. A fórmula T = (D * 500) / 1023 é responsável por converter o valor recebido pelo sensor em temperatura (ºC) e armazenar na variável T, que mais adiante é usada para mostrar o valor da temperatura para o usuário. Para entender esta fórmula, é necessário saber que o microcontrolador Atmega328 do Arduino possui um conversor A/D (Analógico/Digital) de 10 bits de resolução, ou seja, é gerada uma representação digital a partir da grandeza analógica “nível de tensão de saída do sensor”, que vai variar de 0 a 1023. A representação 1023 é o equivalente a 500ºC, faz-se então uma regra de três simples para se chegar na fórmula T = (D * 500) / 1023, onde T é temperatura em graus e D é o sinal analógico detectado pelo sensor.

A pressão (Pascal), altitude (metros) e umidade (porcentagem) são obtidas diretamente com a chamada das funções bmp.readPressure(), bmp.readAltitude() e dht.readHumidity() respectivamente, pois a biblioteca desses sensores já se encarregam de fazer as conversões necessárias.

3.4.1.3 Módulo Transmissor de Radiofrequência

O módulo de radiofrequência é iniciado na estrutura *void setup* com a chamada da função vw_set_tx_pin(TX), onde TX foi declarado como 8, ou seja, o pino de dados do módulo transmissor deve ser conectado ao pino digital 8 do Arduino. A parte do código referente a transmissão dos dados também se encontra dentro da estrutura *void loop*. A estratégia usada para envio dos pacotes de dados é a criação e concatenação dos valores separados por “;” convertidos para um vetor de *bytes*, enviando apenas um pacote contendo todas as informações. A função vw_send é a responsável pelo envio das informações. O Código 3-3 representa a parte do programa responsável por executar o que foi descrito acima.

```
void loop() {
  int D = analogRead(LM35); // leitura analógica do sensor LM35
  int T = (D * 500) / 1023; // Conversão da tensão de entrada para temperatura em graus
  int P = (bmp.readPressure()); // leitura da pressão
  int H = dht.readHumidity(); // leitura da umidade
  int A = (bmp.readAltitude()); // leitura da altitude
  delay(1000);
  //criando vetores para cada variável
```

```
  char Resultadotemp[10];
  itoa(T, Resultadotemp,10);
  char Resultadopressao[10];
  itoa(P, Resultadopressao,10);
  char Resultadoaltitude[10];
  itoa(A, Resultadoaltitude,10);
  char Resultadoumidade[10];
  itoa(H, Resultadoumidade,10);

  //Concatenando os vetores com o separador; para enviar um único pacote
  char TextoEnvio[50] = "";
  strcat(TextoEnvio, Resultadotemp);
  strcat(TextoEnvio, ";");
  strcat(TextoEnvio, Resultadopressao);
  strcat(TextoEnvio, ";");
  strcat(TextoEnvio, Resultadoaltitude);
  strcat(TextoEnvio, ";");
  strcat(TextoEnvio, Resultadoumidade);
  strcat(TextoEnvio, ";");

  digitalWrite(LED, HIGH); // Pisca LED no pino 13 enquanto está transmitindo
  vw_send((uint8_t *)TextoEnvio, strlen(TextoEnvio)); //envio da informação convertida em array de bytes
```

```
  vw_wait_tx(); // Espera o envio da informacao
  delay(1000);
  digitalWrite(LED, LOW); // Apaga o LED no pino 13 quando acaba o envio
}
```

Código 3.3 – Enviando Pacotes de Dados

3.4.2 Estação Receptora

O Sketch da estação receptora será responsável por fazer o reconhecimento das informações transmitidas e a disponibilização destas informações para o usuário, para isso, foi conFigurado o *display* LCD e criada uma em HTML para mostrar os dados em rede local.

3.4.2.1 Inclusão das Bibliotecas

Para o desenvolvimento da estação receptora se fez necessário a utilização das seguintes bibliotecas:

- VirtualWire.h - para o reconhecimento das funções do módulo receptor de radiofrequência;
- LiquidCrystal.h - para facilitar a utilização do *display* LCD;
- SPI.h e *Ethernet*.h - para reconhecimento das funções do *Ethernet Shield*.

O Código 3-4 representa todas as bibliotecas que foram utilizadas para construção da estação receptora.

```
#include <VirtualWire.h> //Biblioteca do modulo de radiofrequência
#include <LiquidCrystal.h> // Biblioteca do display LCD
#include <SPI.h> //Biblioteca do Ethernet Shield
#include <Ethernet.h> // Biblioteca do Ethernet Shield
```

Código 3.4 – Biblioteca da Estação Receptora

3.4.2.2 *Display* LCD

Este *display* é escolhido porque além de ser de baixo custo, ele é compatível com a biblioteca LiquidCrystal do Arduino, o que facilita a programação. O *display* LCD é iniciado na estrutura do *void setup* com a chamada da função lcd.*begin*(16,2). Para escrever na tela do *display* é necessário definir onde o cursor vai começar a escrita com a função lcd.setCursor, em seguida chamamos a função lcd.print para escrever o que desejar. O Código 3-5 mostra a parte do programa onde é conFigurado a escrita das informações meteorológicas no *display* LCD.

```
if(currentMillis - previousMillis > interval) {
      previousMillis = currentMillis;
      lcd.clear();
      switch (info){
        case 0:
        //Escrevendo o valor da temperatura na linha 1 do display LCD
        lcd.setCursor(0,0);
        lcd.print("Temp.: ");
        lcd.print(ValoresRecebidos[0]);
        lcd.print("C");
        //Escrevendo o valor da pressão na linha 2 do display LCD
        lcd.setCursor(0,1);
        lcd.print("Pressao: ");
```

```
    lcd.print(ValoresRecebidos[1]);
    lcd.print("pa");
    break;
  case 1:
    //Escrevendo o valor da altitude na linha 1 do display LCD
    lcd.setCursor(0,0);
    lcd.print("Altitude: ");
    lcd.print(ValoresRecebidos[2]);
    lcd.print(" m")
    //Escrevendo o valor da umidade na linha 2 do display LCD
    lcd.setCursor(0,1);
    lcd.print("Umidade: ");
    lcd.print(ValoresRecebidos[3]);
    lcd.print(" %");
    break;
 }
 info ++;
 if (info > 1)
    info = 0;
```

Código 3.5 – Escrita de Informações no *Display* LCD

O *display* usado no projeto possui 2 linhas e 16 colunas. Na primeira tela será mostrada a temperatura na primeira linha e a pressão na segunda, após mil milissegundos, a tela será limpa e será mostrada a altitude na primeira linha e a umidade na segunda linha. Para alternar as informações no *display* é usada a função millis, que conta milissegundos de forma crescente a partir do momento em que o Arduino é ligado. A cada mil milissegundos (valor atribuído a constante *interval*), o programa limpa o *display* e escreve as informações de temperatura e pressão ou altitude e umidade. Experimentalmente é usado o intervalo de alternação das informações no *display* de mil milissegundos, pois em testes realizados

verificou-se que este é um intervalo de tempo suficiente para a visualização da informação no *display* LCD pelo usuário.

3.4.2.3 *Ethernet Shield*

O *Ethernet Shield* utilizado no projeto é compatível com as bibliotecas SPI.h e *Ethernet*.h do IDE do Arduino, o que facilitou as configurações necessárias dos dados de rede do Arduino, como endereço de IP e endereço MAC. As funções *byte* mac [ ] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; e IPAddress ip(192,168,100,177); definem o endereço MAC e o endereço de IP do Arduino, respectivamente. A conexão *Ethernet* e o serviço *web* são iniciados na estrutura *void setup* do código, com o uso das funções *Ethernet*.*begin*(mac, ip); e server.*begin*(). A página HTML é criada no próprio *Sketch* da estação receptora, dentro da estrutura do *void loop*, para que as informações sejam sempre atualizadas. A função client.println("Refresh: 5"); define que a página HTML será atualizada a cada 5 segundos, ou seja, as informações mostradas no *display* LCD têm um intervalo de atualização mais rápido.

O Código 3-6 representa a parte do programa onde é construída a página HTML do projeto.

```
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("Refresh: 5");  // Atualiza a página automaticamente a cada 5 segundos
client.println();
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.print("<head><title>Projeto Final</title></head> ");
client.print("<body bgcolor=#EEE><center><br><br><table border=1px> ");
```

```
client.print("<tr><th bgcolor=#FFF align=left><font size=7>UniCEUB - Centro Universitário de Brasilia</font><br> ");
client.print("<br><font size=5>Joao Paulo Rosito<br>Engenharia da Computacao</font></tr></th> ");
client.print("<tr><th bgcolor=#FFF><font size=6>Estacao Meteorológica Automatizada</font></tr></th><tr><th bgcolor=#FFF><center><table> ");
client.print("<tr><th align=right><font size=7>Temperatura:  </font></th>  <th align=left><font size=7 color=#1E90FF> ");
client.print(ValoresRecebidos[0]);
client.print(" *C</font></th> </tr> ");
client.print("<tr><th align=right><font size=7>Pressao Atmosferica:  </font></th>  <th align=left><font size=7 color=#1E90FF> ");
client.print(ValoresRecebidos[1]);
client.print(" pa</font></th> </tr> ");
client.print("<tr><th align=right><font size=7>Altitude Real:  </font></th>  <th align=left> <font size=7 color=#1E90FF> ");
client.print(ValoresRecebidos[2]);
client.print(" m</font></th> </tr> ");
client.print("<tr><th align=right><font size=7>Umidade Rel.:</font></th>  <th align=left> <font size=7 color=#1E90FF> ");
client.print(ValoresRecebidos[3]);
client.print(" %</font></th> </tr> ");
client.print("</table></center></th></tr></table></center></body> ");

client.println("</html>");
```

Código 3.6 – Construção da Página HTML

3.4.2.4 Módulo Receptor de Radiofrequência

O módulo receptor é conFigurado com a chamada da função vw_set_rx_pin(RX), sendo RX definido como 8, ou seja, o pino de dados do módulo receptor deve ser conectado ao pino digital 8 do Arduino. O módulo receptor é

iniciado com a função vw_rx_start, dentro da estrutura do *void setup*. A função vw_get_message é responsável por receber a mensagem. É recebida uma mensagem em vetor de *bytes* contendo todas as informações (temperatura, pressão, altitude e umidade) separadas por “;”, que posteriormente será tratada para separar os dados, conforme representado no Código 3-7.

```
for (int i = 0; i < buflen; i++) { // Percorre o vetor de byte (buf) byte a byte

    if(buf[i]==';'){ //Verifica se o byte lido é um separador “;“

     conta=conta+1; // Incrementa a posição onde o valor será gravado no vetor ValoresRecebidos

    } else {

     ValoresRecebidos[conta]+=char(buf[i]); //Grava o valor do byte recebido na posição atual do vetor ValoresRecebidos

    }

}
```

Código 3.7 – Tratamento das informações recebidas pelo módulo receptor

Foi criada uma variável do tipo vetor com 4 posições, onde cada posição armazena um valor do pacote recebido.

- ValoresRecebidos [0] que será a temperatura;
- ValoresRecebidos [1] que será a pressão;
- ValoresRecebidos [2] que será a altitude;
- ValoresRecebidos [3] que será a umidade;

A variável ValoresRecebidos será utilizada para exibir os parâmetros meteorológicos para o usuário no *display* LCD e na página HTML.

3.5 Desenvolvimento da Placa de Circuito Impresso

A construção do protótipo em placa de circuito impresso é a etapa final do desenvolvimento do projeto. Os desenhos das placas da estação transmissora e receptora foram feitos com o auxílio do *software* Proteus. Proteus é um suíte que agrega um *software* para simulação de circuitos eletrônicos (ISIS) e um *software* para desenho do circuito impresso (ARES).

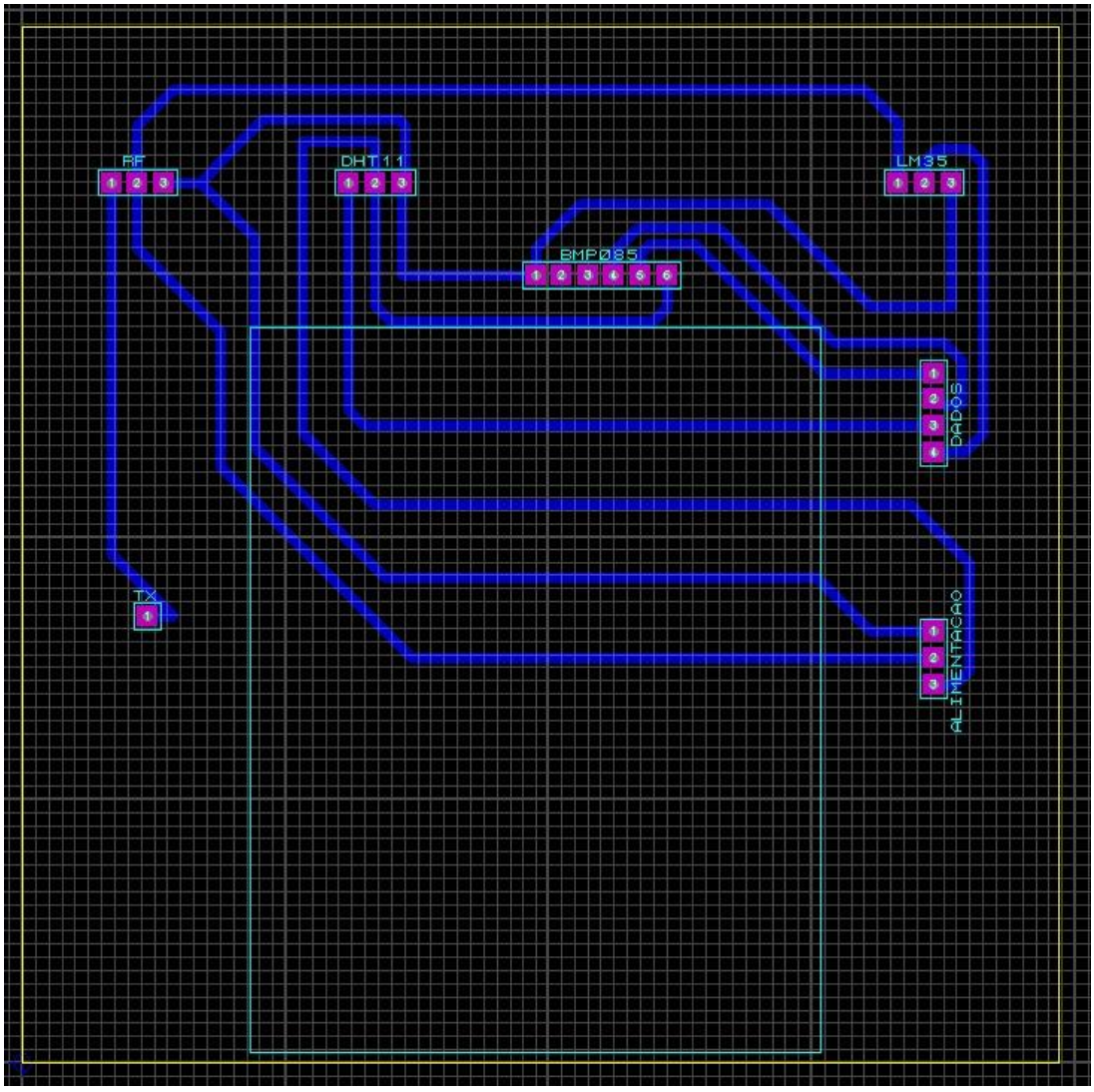

Figura **3-10 – Desenho do Circuito Impresso da Placa Transmissora (Fonte: Autor)**

A Figura 3-10 ilustra o desenho do circuito que foi impresso na placa de fenolite. As linhas em azul são as trilhas de cobre do circuito e os quadrados em rosa indicam onde serão soldados os conectores para encaixe dos sensores.

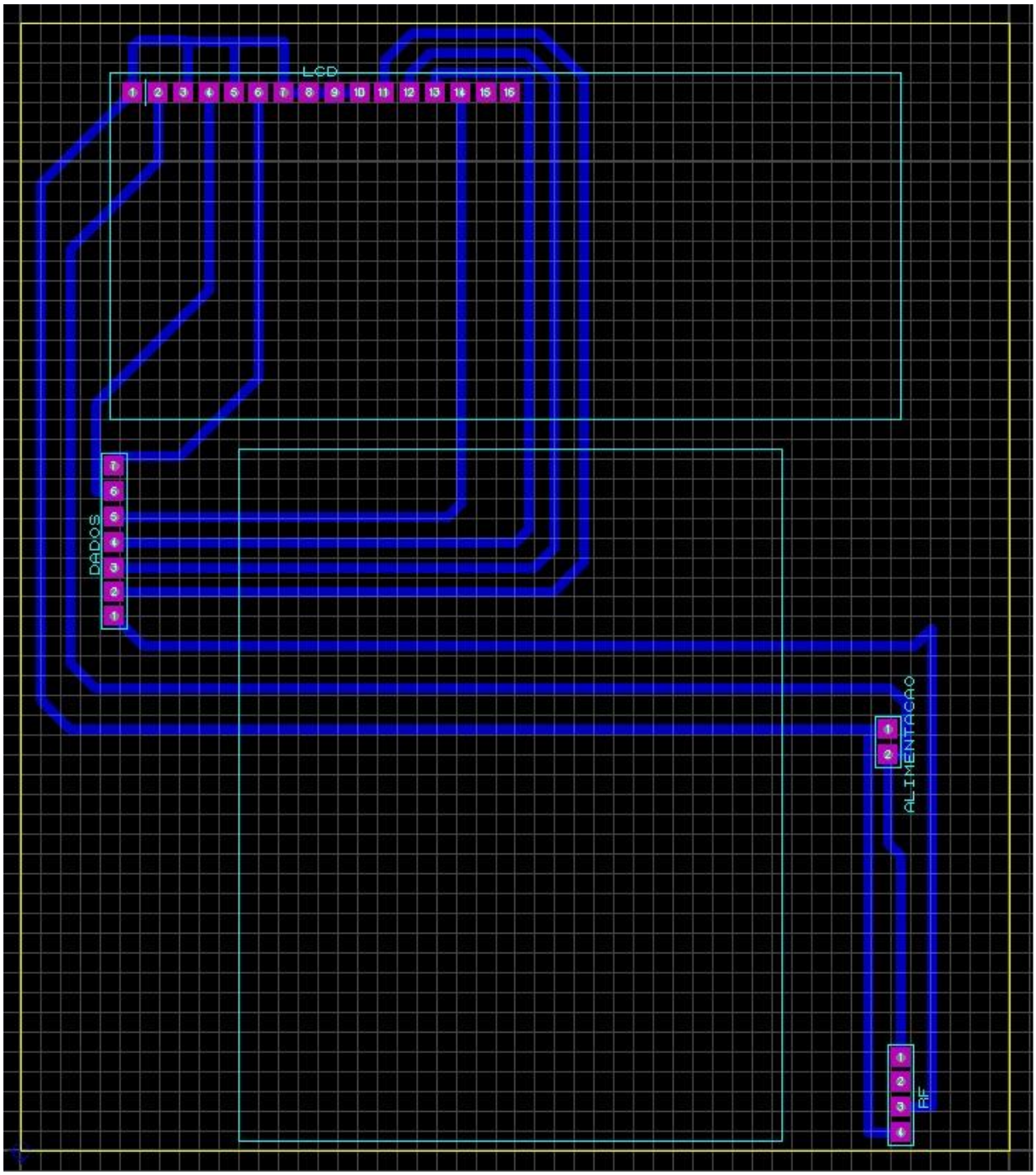

Figura **3-11 – Desenho do Circuito Impresso para Placa Receptora**
(Fonte: Autor)

A Figura 3-11 ilustra o desenho do circuito que foi impresso na placa de fenolite.

Os esquemas ilustrados pelas Figuras 3.10 e 3.11 foram impressos em papel fotográfico para que fossem passados para as placas de fenolite, onde foi montado o protótipo final do projeto. Neste processo, foram fixadas as impressões no papel fotográfico às placas e em seguida foi feita a termo transferência, que consiste em aplicar calor sobre as folhas de papel fotográfico até que a tinta seja transferida para a placa. Feito isso, foi aplicado um ácido à placa, para que corroa o cobre da placa deixando apenas a parte coberta pela tinta do papel fotográfico. Ao limpar as placas todos os sensores e componentes foram instalados.

A Figura 3-12 mostra o protótipo da estação transmissora montado com todos os componentes instalados.

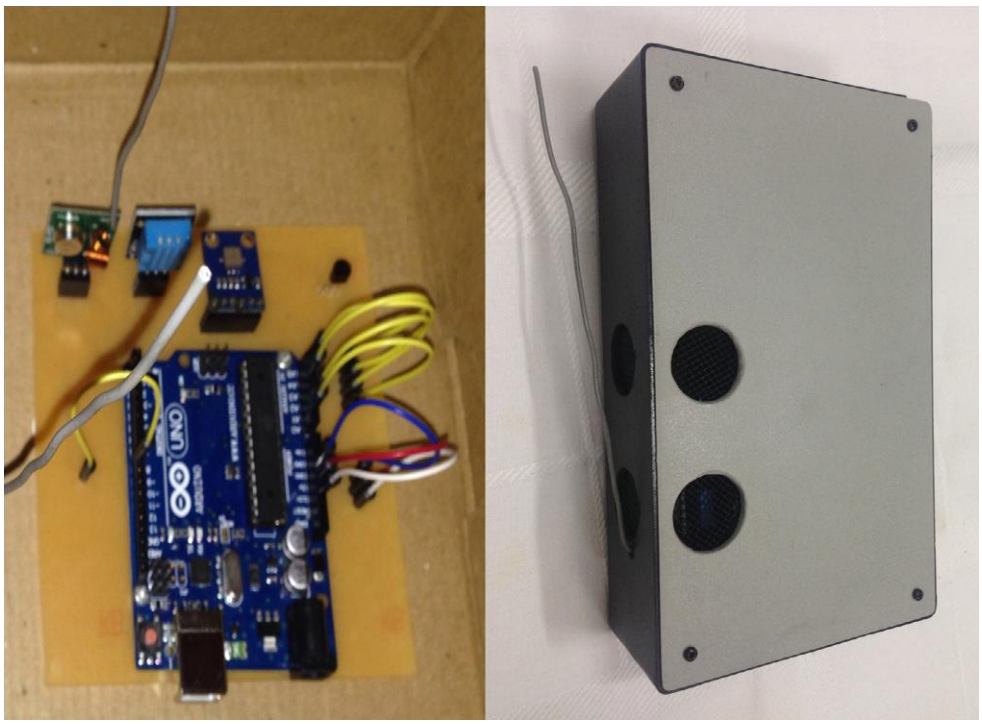

Figura **3-12 – Protótipo final da estação transmissora**
(Fonte: Autor)

A Figura 3-13 mostra o protótipo da estação receptora montado com todos os componentes instalados.

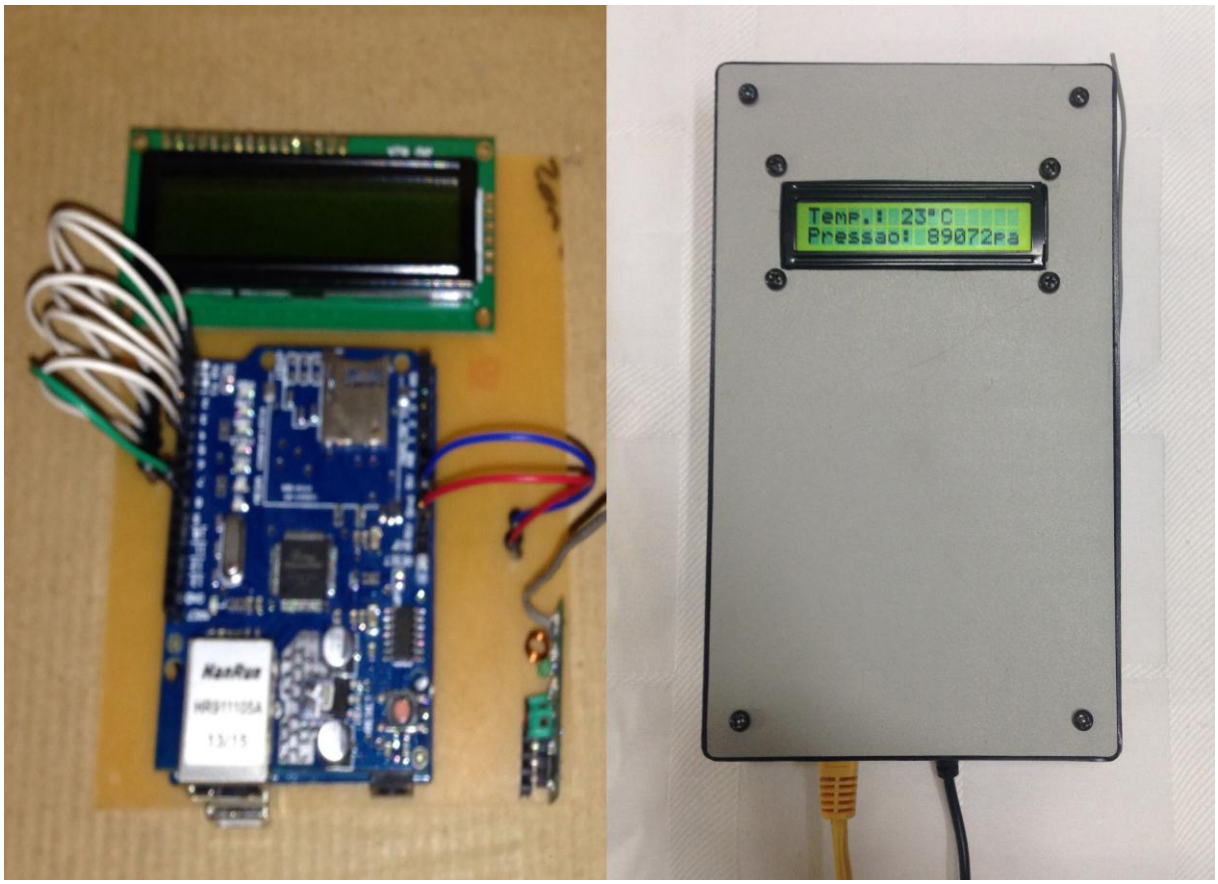

Figura **3-13 – Protótipo final da estação receptora**
(Fonte: Autor)

CAPÍTULO 4 TESTES E RESULTADOS

Neste capítulo serão apresentadas as descrições dos testes realizados de todos os componentes utilizados no projeto, com o objetivo de avaliar o desempenho e verificar se o mesmo atende ao que se foi proposto. Serão testados se todos os sensores utilizados apresentam resultados satisfatórios e se a comunicação entre as estações será feita de forma eficiente, com respostas rápidas e sem falha.

4.1 Cenários de testes

4.1.1 Cenário 1

O objetivo deste cenário é testar o sensor de temperatura LM35. Para controlar a variação da temperatura é utilizado um ferro de solda. Ao encostar o ferro de solta no sensor, em cerca de 2 segundos a temperatura já se mostrava alterada no *display* LCD.

Foi visto no referencial teórico que a tensão de saída neste sensor varia 10 mV para cada grau variado. É criado um gráfico (Gráfico 4-1) para constatar se neste sistema esta variação está acontecendo de fato de forma linear. Foram feitas 10 medições para termos um resultado seguro e detalhado.

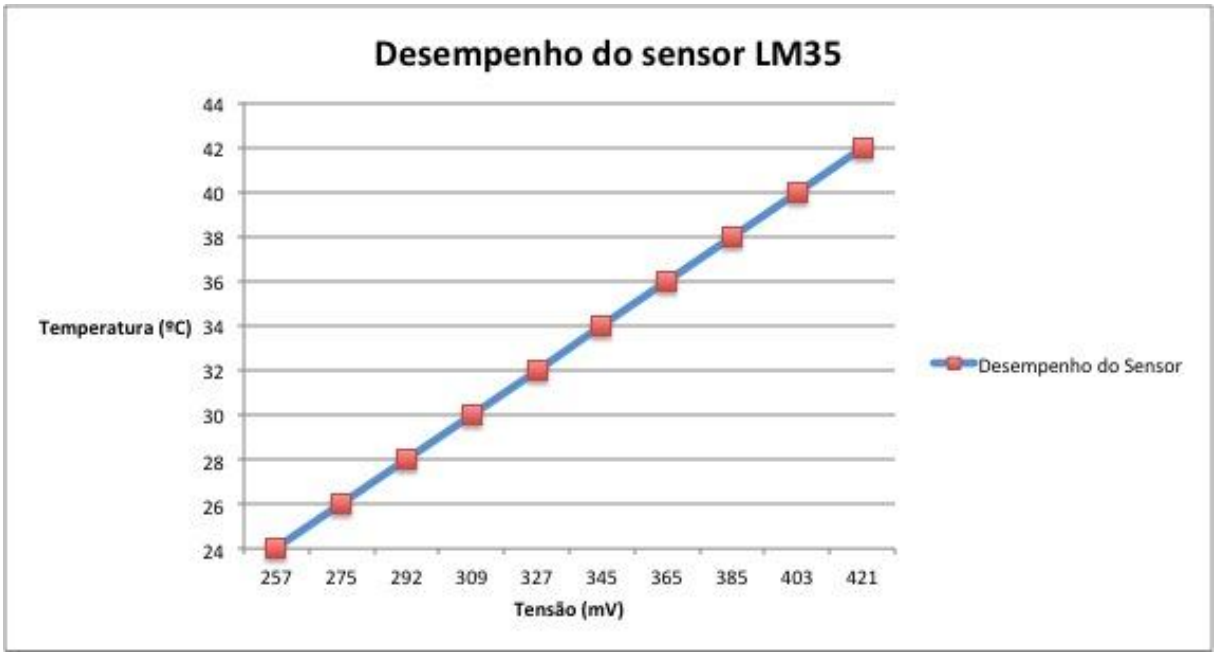

Gráfico 4-1 - Desempenho do sensor de temperatura LM35
(Fonte: Autor)

A média de variação da tensão entre uma medição e outra é de 18 mV e o intervalo de temperatura é de 2 ºC, ou seja, 9 mV para cada grau. Como visto no capítulo 2, o sensor LM35 tem uma margem de erro de, no máximo, 3/4ºC para mais ou para menos, 1 mV representa 1/10 ºC, isto é, está dentro das margens especificadas pelo fabricante. Para realização do teste, é utilizado um multímetro e pôde-se concluir que esta variação está acontecendo de forma linear e que o sensor atende aos requisitos do projeto.

4.1.2 Cenário 2

Neste cenário é testado o sensor de umidade DHT11. Foi utilizado um umidificador próximo ao sensor, para simular a variação da umidade relativa local, possibilitando a realização dos testes. O Gráfico 4-2 representa a variação da umidade relativa apresentada pelo sensor em um intervalo de 20 segundos.

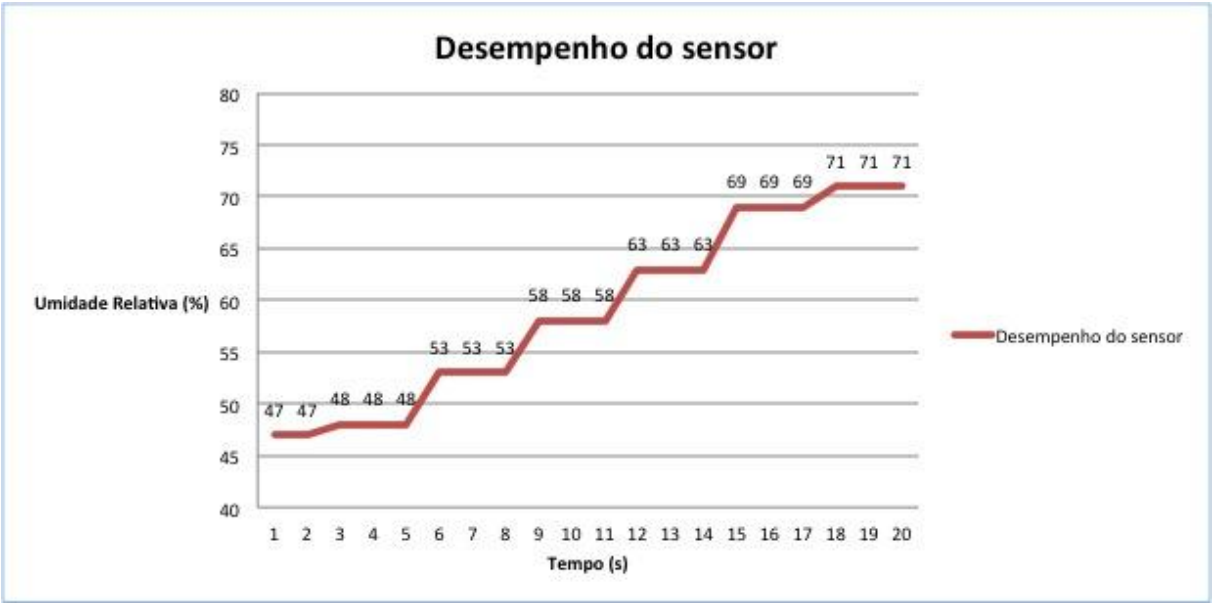

Gráfico 4-2 - Desempenho do sensor DHT11
(Fonte: Autor)

O sensor foi testado também em um dia de chuva, apresentando umidade de 87% de UR. Este resultado é comparado com a amostragem feita pelo INMET em Brasília, no mesmo momento, que é de 93% de UR, ou seja, o protótipo apresenta um resultado aceitável, tendo em vista que o protótipo está coletando informações de uma região diferente das que foram coletadas pelos sensores do INMET.

4.1.3 Cenário 3

No cenário 3 é testado o sensor de pressão BMP085. O teste realizado para este sensor foi analisar o resultado apresentado e verificar se é um resultado aceitável, pois não há um ambiente de teste que permita o controle da pressão atmosférica local. A pressão atmosférica apresentada pelo sensor é de 89088 Pa (pascal) ou 890,8 hPa (hecto pascal). No site do INMET (Instituto Nacional de Meteorologia) a pressão atmosférica apresentada no mesmo instante em uma estação meteorológica instalada também em Brasília, é de 885,5hPa. Levando em consideração que o teste realizado no sensor foi feito em um lugar diferente das

medições do INMET, conclui-se que o sensor está apresentando os resultados esperados.

A Figura 4-1 mostra o valor de pressão coletado pela estação meteorológica do projeto ao lado do valor de pressão coletado pela estação meteorológica do INMET.

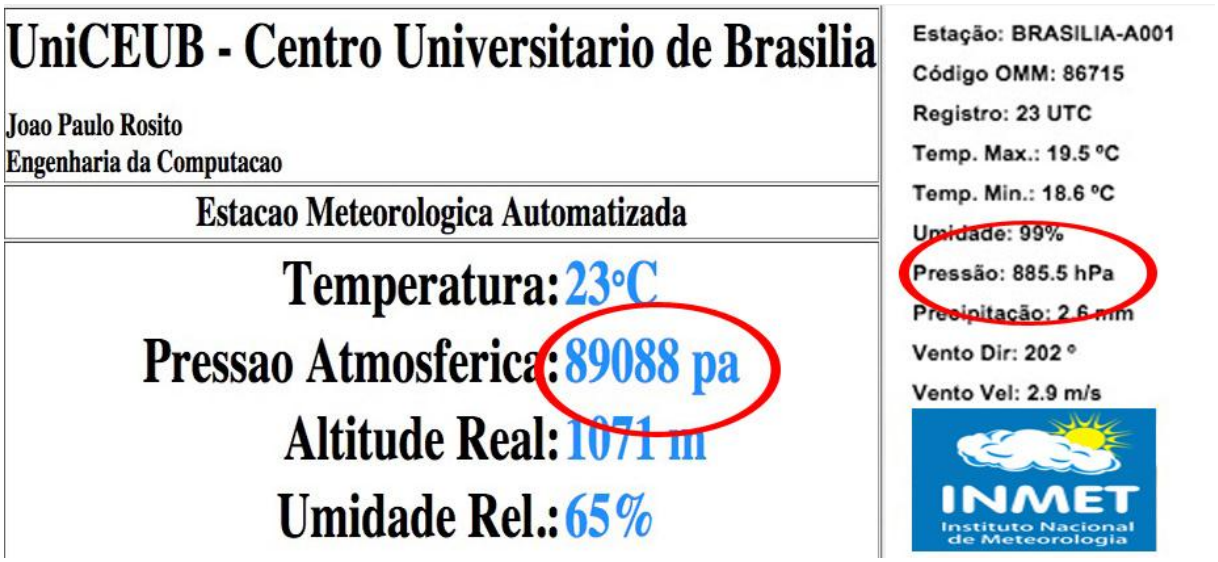

Figura **4-1 - Teste de pressão atmosférica**
(Fonte: Autor)

Para o teste da altitude, é medida a altitude no nível do solo, que apresenta 1071 metros, e posteriormente é suspensa a estação transmissora a 4 metros do nível do solo, apresentando altitude de 1075 metros. O resultado dos testes de altitude também foram satisfatórios, apresentando os resultados esperados.

4.1.4 Cenário 4

No cenário 4 é testado o desempenho do módulo de radiofrequência. Para a realização da análise da frequência emitida pelo módulo transmissor de rádio, foi utilizado o Analisador de Espectro Agilent NB9320B, com a faixa de operação entre 410 MHz e 440 MHz, para que se tenha melhor visualização do

sinal no analisador. O módulo transmissor é posicionado a 1 metro do analisador e do módulo receptor. A frequência apresentada pelo analisador de espectro é de 433,93 MHz, este resultado está aceitável, tendo em vista que a frequência do módulo RF especificada pelo fabricante é de 433 MHz. A Figura 4-2 ilustra o sinal capturado pelo analisador.

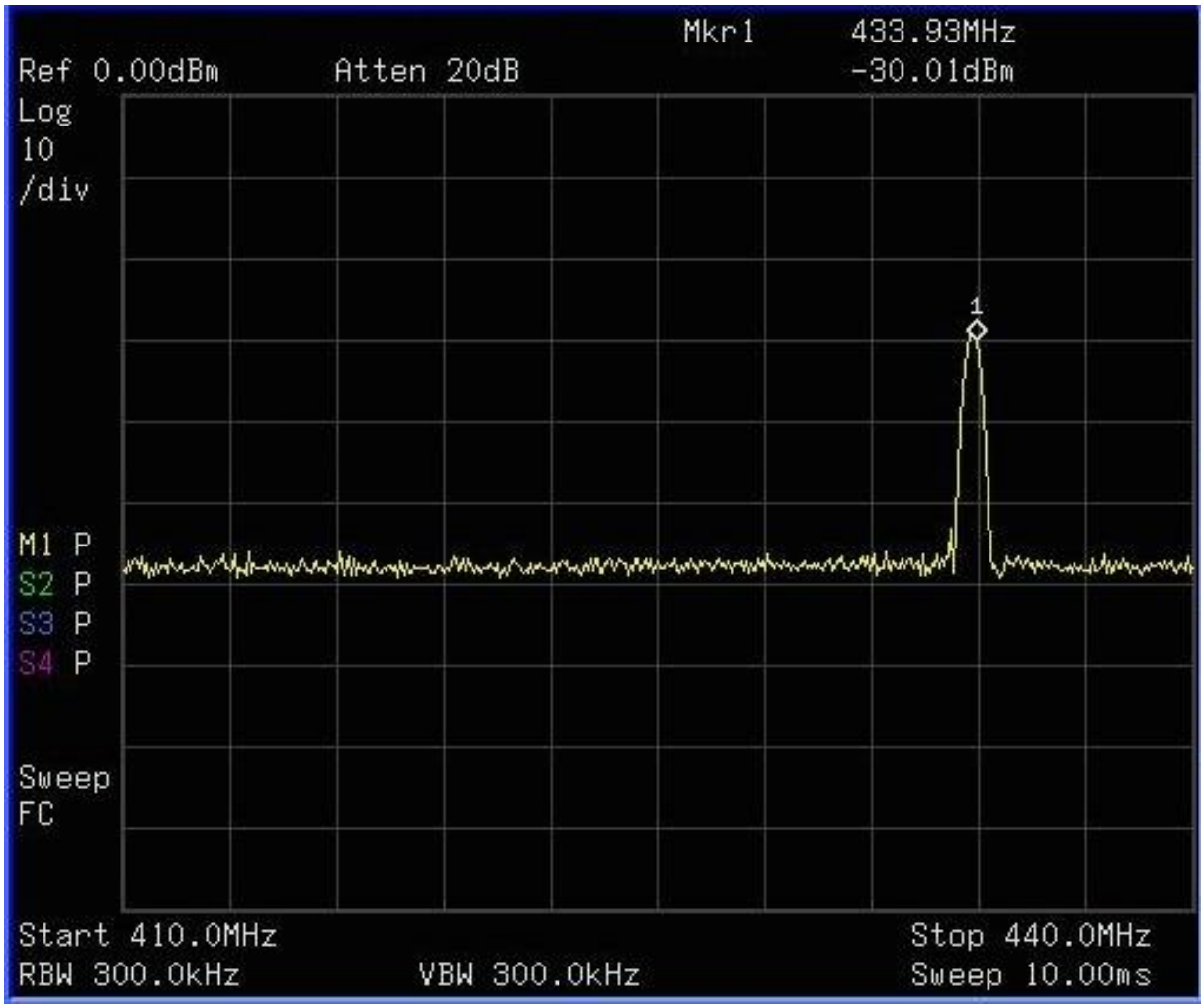

Figura **4-2 - Analisador de espectro**
(Fonte: Autor)

Ainda com o auxilio do analisador, é possível observar também a potência emitida pelo módulo transmissor que é de aproximadamente -30,01 dBm. Aplicando a relação logarítmica abaixo, têm-se a potência absoluta emitida pelo transmissor, dada em milliwatts:

$$PdBm = 10 \log 10 \left(\frac{PmW}{1mW}\right)$$

ou

$$P_{(mW)} = 1mW \cdot 10^{(P(dBm)/10)}$$

Obtêm-se o resultado de $9{,}98 \times 10^{-4}$ miliwatts de potência absoluta com a aplicação da fórmula acima. Outro teste é realizado neste cenário, mas desta vez com o módulo transmissor distanciado do receptor até o ponto máximo antes que se perca o sinal. A Figura 4-3 representa o sinal captado pelo analisador de espectro para este teste.

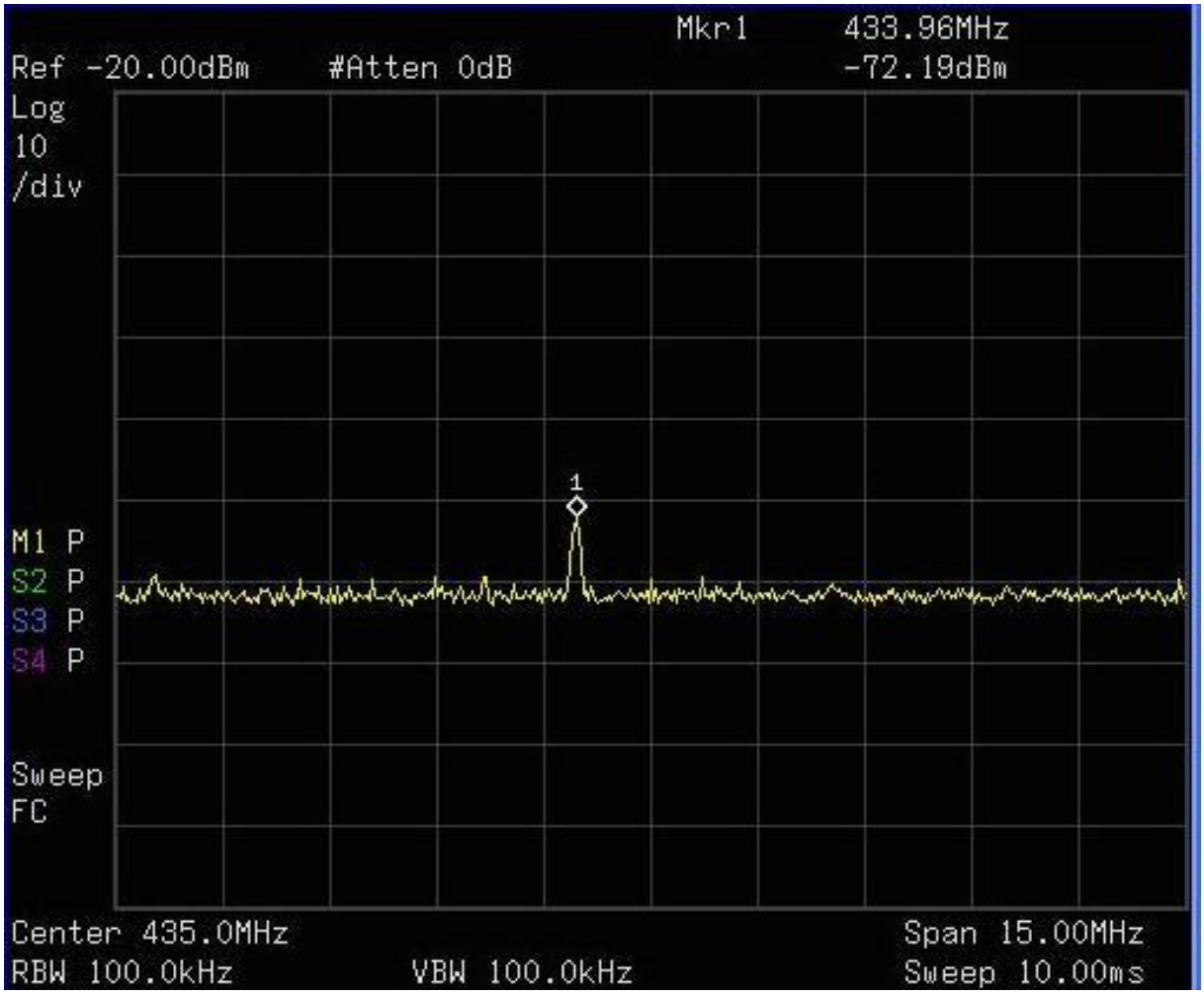

Figura **4-3 - Analisador de espectro**
(Fonte: Autor)

Neste caso a potência captada pelo analisador é de -72,19 dBm, que nos fornece uma potência absoluta de $6{,}04\times10^{-8}$ milliwatts à 60 metros de distância, o que significa dizer que para uma potência de $6{,}04\times10^{-8}$ milliwatts, este protótipo pode nos dar um alcance de até 60 metros.

4.1.5 Cenário 5

Este cenário foi reservado para o teste do monitoramento das informações em rede local. Foram feitos 2 acessos simultâneos em 2 pontos de rede diferentes. A página HTML criada foi aberta com sucesso para os dois pontos. Para realização do acesso às informações em rede local, foi necessário abrir um navegador *web* e digitar o endereço de IP do Arduino (192.168.100.177) na barra de navegação. A página HTML tem um intervalo de atualização mais lenta do que o *display* LCD, pois está programada para atualizar a cada 5 segundos, quando que no *display* as informações são atualizadas a cada 3 segundos, porém para o projeto esta diferença é desprezível. A Figura 4-4 ilustra a página HTML criada para o monitoramento das informações.

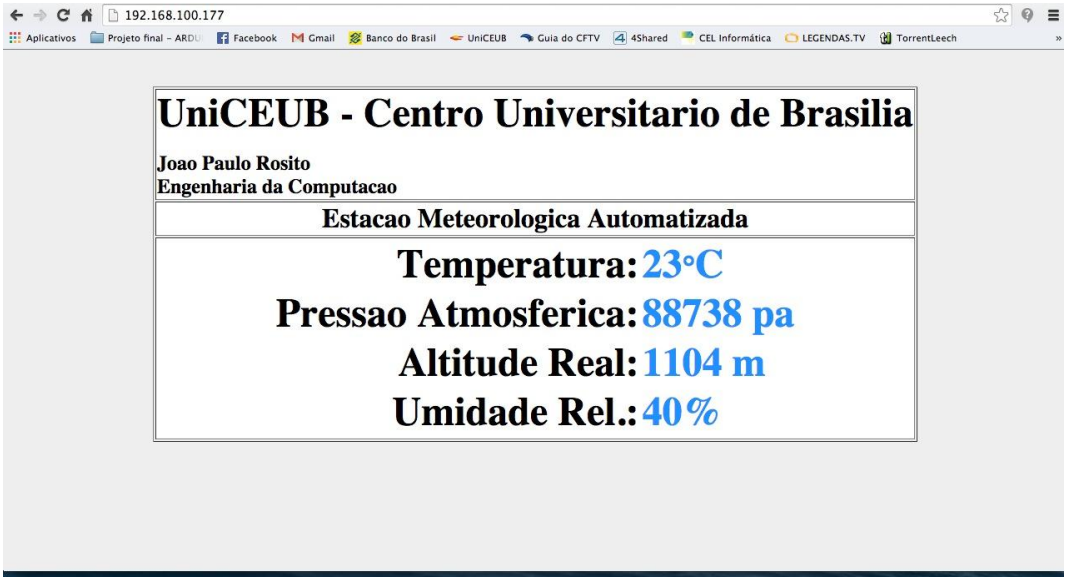

Figura **4-4 - Página HTML de monitoramento das informações (Fonte: Autor)**

CAPÍTULO 5 CONCLUSÃO

Tendo como ponto focal as áreas urbanas carentes de políticas públicas adequadas, com riscos de sofrerem desastres naturais, especialmente aqueles provocados por situações meteorológicas adversas, identificou-se a necessidade de utilização de soluções tecnológicas para auxiliar essas comunidades com a informação prévia sobre a ocorrência desses eventos. Buscando minimizar suas consequências, em razão da vulnerabilidade aos acidentes gerados por períodos de chuvas mais intensas, que atingem, principalmente, favelas e loteamentos irregulares das encostas de morros urbanos.

Assim, este projeto teve por objetivo o desenvolvimento de um protótipo capaz de auxiliar a previsão de alterações climáticas, a fim de tornar possível um alerta prévio à população local, para que seja feita uma preparação ou evacuação dos locais de risco, caso necessário, evitando então, os desastres citados neste estudo.

O objetivo foi alcançado de forma satisfatória. Foi construída a Estação Meteorológica Automatizada Remota de fácil instalação, com duas estações (Transmissora e Receptora) que apresentaram resultados positivos de desempenho. Os dados meteorológicos de temperatura, pressão e umidade foram coletados e informados ao usuário em tempo e precisão de resposta que permitem que sejam tomadas ações antecipadas de alerta e evacuação dos locais monitorados, atendendo ao que foi proposto. Os testes realizados comprovam a viabilidade econômica e tecnológica do projeto.

A construção do protótipo proposto neste trabalho foi possível com os conhecimentos adquiridos ao longo do curso de Engenharia da Computação, fundamentais para o desenvolvimento do projeto. Foram aplicados conceitos aprendidos em disciplinas como linguagem técnica de programação, circuitos eletrônicos, sistemas de comunicação, redes de computadores, além dos que foram estudados no referencial teórico citado no capítulo 2.

Linguagem técnica de programação foi importante para o desenvolvimento do *software* das estações, que foi construído em linguagem *Wiring* (baseado em C); conhecimentos de circuitos eletrônicos fundamentaram o entendimento e a montagem dos componentes eletrônicos utilizados na construção do protótipo; o estudo de sistemas de comunicação auxiliou na implementação da forma de comunicação escolhida para o projeto, que é o envio das informações meteorológicas por radiofrequência; conceitos aplicados sobre redes de computadores foram fundamentais para montagem e configuração da rede local instalada na estação receptora, para o monitoramento das informações de qualquer ponto da rede.

O principal desafio do trabalho foi estabelecer a comunicação entre as estações transmissora e receptora, de forma eficiente e que apresentasse um bom desempenho. Esta dificuldade foi superada pela solução tecnológica aplicada, a comunicação foi estabelecida e apresentou resultados satisfatórios e que atendem aos requisitos estabelecidos no escopo do projeto.

5.1 SUGESTÕES PARA TRABALHOS FUTUROS

Como sugestão de trabalhos futuros, pode-se implementar um *software* mais robusto integrado com banco de dados para guardar um histórico das variações meteorológicas tornando possíveis análises mais complexas e precisas das variações climáticas.

Na parte do projeto responsável pela comunicação entre as estações, é utilizado um módulo de radiofrequência de 433 MHz, que, com alimentação de 5 V, teve um alcance de 60 metros sem barreiras. Como sugestão para trabalhos futuros pode-se utilizar transmissor e receptor de rádio, com frequências maiores, na casa dos giga hertz, para que se tenha maiores alcances de sinal.

Para transmissão dos dados pode-se estudar outras formas de comunicação, como envio das informações por 3G ou 4G, o que torna possível acesso das informações, não só da estação base, mas como de qualquer lugar que tenha acesso à internet.

Estas sugestões contribuirão para realização de novos estudos no sentido de aprimorar tecnologias voltadas para prevenção de catástrofes socioambientais, deixando a vida de muitas pessoas mais segura.

REFERÊNCIAS

LADYADA.NET. (2012). Acesso em 15 de setembro de 214, disponível em http://www.ladyada.net/learn/sensors

TEXAS INSTRUMENTS. (2013). Acesso em 15 de setembro de 2014, disponível em http://www.ti.com/lit/ds/symlink/lm35.pdf

FILIPEFLOP. (2014). Acesso em 13 de Agosto de 2014, disponível em http://blog.filipeflop.com

GATEWAYIT. (2014). Acesso em 2 de outubro de 2014, disponível em gtewayit: http://gatewayit.com.br

INMET. (2014). Acesso em 20 de 10 de 2014, disponível em Instituto Nacional de Meteorologia: http://www.inmet.gov.br/portal/

PROTOSTACK. (2014). Acesso em 19 de outubro de 2014, disponível em http://www.protostack.com/

ALVARENGA, B., & MAXIMO, A. (2009). *Fisica* (Vol. Unico). Ática.

ARDUINO E CIA. (s.d.). Acesso em 20 de setembro de 2014, disponível em http://www.arduinoecia.com.br

ARDUINO.CC. (s.d.). Acesso em 15 de Agosto de 2014, disponível em http://arduino.cc.com

CARVALHO, C. S., & GALVAO, T. (2006). *Prevenção de Riscos de Deslizamentos em Encostas.* Brasília: Cities Alliance.

DORF, R. C., & BISHOP, R. H. (2013). *Sistemas de controle modernos* (Oitava ed.).

MCROBERTS, M. (2011). *Arduino Básico.* São Paulo, SP: Novatec.

MONK, S. (2013). *Programação com arduino - começando com sketches.* (Bookman, Ed.) Porto Alegre.

SENSORTEC, B. (2009). *sparkfun*. Acesso em 16 de outubro de 2014, disponível em https://www.sparkfun.com/datasheets/Components/General/BST-BMP085-DS000-05.pdf

THOMANZINI, D., & ALBUQUERQUE, P. U. (2008). *Sensores industriais - Fundamentos e Aplicações.* (E. Érica, Ed.) São Paulo.

TOFFOLI, L. (11 de 2014). Pressao Atmosferica.

WAIHUG.NET. (s.d.). Acesso em 3 de outubro de 2014, disponível em waihung.net/arduino-ethernet-shield-r3

WETHERLY, K. (2008). Acesso em 20 de outubro de 2014, disponível em How Stuff Works: science.howstuffworks.com

YOUNG, H. D., & FREEDAN, R. A. (2012). *Física I - mecânica* (Vol. 12). (pearson, Ed.) São Paulo.

APÊNDICE A - Estação transmissora

```
/*---------------------------------------------------------------------------
/ Projeto Final
/ Joao Paulo Rosito
/ Estacao Meteorologica Automatizada Remota
/ Centro Universitario de Brasilia
/
/       ESTACAO TRANSMISSORA
/
/---------------------------------------------------------------------------*/


#include <VirtualWire.h> // Biblioteca do modulo de Radiofrequencia
#include <Wire.h> // Biblioteca do barometro
#include <Adafruit_BMP085.h> // Biblioteca do barometro
#include "DHT.h" //Biblioteca do sensor de umidade


#define DHTPIN A2 // Define o pino de dados do sensor de umidade
#define DHTTYPE DHT11 // Define o tipo do sensor de umidade DH11

Adafruit_BMP085 bmp; //Declarando barometro
DHT dht(DHTPIN, DHTTYPE); // Declarando o sensor de umidade

const int LM35 = A1; // Definindo sensor LM35 de temperatura na porta analogica A1
const int LED = 13; // Define led da porta 13
```

```
const int TX = 8; // Atribui o valor da variavel TX que sera usada como pino de dados do modulo RF

void setup() {

  Serial.begin(9600); // Starta o serial monitor

  vw_set_tx_pin(TX); // Define o pino de dados do modulo de Radiofrequencia

  vw_set_ptt_inverted(true); // Requerido para DR3100

  vw_setup(2000); // Bits por segundos

  if (!bmp.begin()) { //funcao que testa se o sensor de pressao foi encontrado

        Serial.println("Nao foi possivel encontrar o sensor barometrico, verifique as ligacoes!");

        while (1) {}

  }

}


void loop() {

  int D = analogRead(LM35); // leitura analogica do sensor LM35

  int T = (D * 500) / 1023; // Conversao da tensao de entrada para temperatura em graus

  int P = (bmp.readPressure()); // leitura da pressao

  int H = dht.readHumidity(); // leitura da umidade

  int A = (bmp.readAltitude()); // leitura da altitude

  delay(1000);

  //criando vetores para cada variavel

  char Resultadotemp[10];

  itoa(T, Resultadotemp,10);

  char Resultadopressao[10];

  itoa(P, Resultadopressao,10);

  char Resultadoaltitude[10];

  itoa(A, Resultadoaltitude,10);

  char Resultadoumidade[10];
```

```
  itoa(H, Resultadoumidade,10);

  //Concatenando os vetores com o separador ; para enviar um unico pacote
  char TextoEnvio[50] = "";
  strcat(TextoEnvio, Resultadotemp);
  strcat(TextoEnvio, ";");
  strcat(TextoEnvio, Resultadopressao);
  strcat(TextoEnvio, ";");
  strcat(TextoEnvio, Resultadoaltitude);
  strcat(TextoEnvio, ";");
  strcat(TextoEnvio, Resultadoumidade);
  strcat(TextoEnvio, ";");


  digitalWrite(LED, HIGH); // Pisca LED no pino 13 enquanto esta transmitindo
  vw_send((uint8_t *)TextoEnvio, strlen(TextoEnvio)); //envio da informacao
  vw_wait_tx(); // Espera o envio da informacao
  delay(1000);
  digitalWrite(LED, LOW); // Apaga o LED no pino 13 quando acaba o envio
}
```

APÊNDICE B - Estação receptora

```
/*---------------------------------------------------------------------------
/ Projeto Final
/ Joao Paulo Rosito
/ Estacao Meteorologica Automatizada Remota
/ Centro Universitario de Brasilia
/
/       ESTACAO RECEPTORA
/
/---------------------------------------------------------------------------*/


#include <VirtualWire.h> //Biblioteca do modulo de Radiofrequencia
#include <LiquidCrystal.h> // Biblioteca do display LCD
#include <SPI.h> //Biblioteca do Ethernet Shield
#include <Ethernet.h> // Biblioteca do Ethernet Shield


byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,100,177);
EthernetServer server(80);


LiquidCrystal lcd(1, 0, 5, 4, 3, 2);
const int LED = 13;
const int RX = 8;
```

```
long previousMillis = 0;        // will store last time LED was updated
long interval = 1000;           // interval at which to blink (milliseconds)
int info=0;


void setup() {
 // Open serial communications and wait for port to open:
  lcd.begin(16,2);
  // Inicializando E/S do receptor
  vw_set_rx_pin(RX);
  vw_set_ptt_inverted(true); // Requerido para DR3100
  vw_setup(2000); // Bits por segundo
  vw_rx_start(); // Inicia a recepcao
  //Serial.begin(9600);
   while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }



  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}
```

```
void loop() {
  unsigned long currentMillis = millis();
  uint8_t buf[VW_MAX_MESSAGE_LEN];
  uint8_t buflen = VW_MAX_MESSAGE_LEN;
  // listen for incoming clients
  if (vw_get_message(buf, &buflen)) { // Recebe o vetor de bytes enviados pelo transmissor
    digitalWrite(LED, HIGH); // Pisca LED no pino 13 se receber a mensagem integra

    String ValoresRecebidos[4] = {"","","",""}; // Declara vetor do tipo string para armazenar os valores enviados
    int conta; // variavel criada para verificar o separador ";"
    conta=0; // Determina a primeira posicao do vetor ValoresRecebidos para comecar a receber a informacao

    for (int i = 0; i < buflen; i++) { // Percorre o vetor de byte (buf) recebido byte a byte
      if(buf[i]==';'){ // Verifica se o byte lido eh um separador ";"
        conta=conta+1; // Incrementa a posicao onde o valor sera gravado no vetor de ValoresRecbidos
      } else {
        ValoresRecebidos[conta]+=char(buf[i]); //Grava o valor do byte recebido na posicao atual do vetor ValoresRecebidos, concatenando com a informacao existente nesta posicao.
      }
    }
```

```
if(currentMillis - previousMillis > interval) {
    previousMillis = currentMillis;
    lcd.clear();

    switch (info){
      case 0:
      lcd.setCursor(0,0);
      lcd.print("Temp.: ");
      lcd.print(ValoresRecebidos[0]);
      lcd.print("C");

      lcd.setCursor(0,1);
      lcd.print("Pressao: ");
      lcd.print(ValoresRecebidos[1]);
      lcd.print("pa");
      break;

    case 1:
      lcd.setCursor(0,0);
      lcd.print("Altitude: ");
      lcd.print(ValoresRecebidos[2]);
      lcd.print(" m");
```

```
        lcd.setCursor(0,1);
        lcd.print("Umidade: ");
        lcd.print(ValoresRecebidos[3]);
        lcd.print(" %");
        break;
    }
    info ++;
    if (info > 1)
      info = 0;
}



EthernetClient client = server.available();
if (client)
 {
   boolean currentLineIsBlank = true;
   while (client.connected())
    {
      if (client.available())
        {
          char c = client.read();
          Serial.write(c);


          if (c == '\n' && currentLineIsBlank)
```

```
            {
              client.println("HTTP/1.1 200 OK");
              client.println("Content-Type: text/html");
              client.println("Connection: close");  // conexão sera fechada apos completar a
resposta
            client.println("Refresh: 5");  // Atualiza a página HTML a cada 5 segundos
              client.println();
              client.println("<!DOCTYPE HTML>");
              client.println("<html>");



              client.print("<head><title>Projeto Final</title></head> ");
              client.print("<body bgcolor=#EEE><center><br><br><table border=1px> ");
              client.print("<tr><th bgcolor=#FFF align=left><font size=7>UniCEUB - Centro
Universitario de Brasilia</font><br> ");
              client.print("<br><font size=5>Joao Paulo Rosito<br>Engenharia da
Computacao</font></tr></th> ");
              client.print("<tr><th bgcolor=#FFF><font  size=6>Estacao Meteorologica
Automatizada</font></tr></th><tr><th bgcolor=#FFF><center><table> ");


              client.print("<tr><th align=right><font size=7>Temperatura:      </font></th>    <th
align=left><font size=7 color=#1E90FF> ");
              client.print(ValoresRecebidos[0]);
              client.print("<code>&deg;</code>"); // inserir o simbolo de grau
              client.print(" C</font></th> </tr> ");
```

```
            client.print("<tr><th align=right><font size=7>Pressao Atmosferica:
        </font></th>    <th align=left><font size=7 color=#1E90FF> ");

            client.print(ValoresRecebidos[1]);

            client.print(" pa</font></th> </tr> ");


            client.print("<tr><th align=right><font size=7>Altitude Real:      </font></th>    <th
align=left> <font size=7 color=#1E90FF> ");

            client.print(ValoresRecebidos[2]);

            client.print(" m</font></th> </tr> ");


            client.print("<tr><th align=right><font size=7>Umidade Rel.:       </font></th>  <th
align=left> <font size=7 color=#1E90FF> ");

            client.println(ValoresRecebidos[3]);


            client.print(" %</font></th> </tr> ");


            client.print("</table></center></th></tr></table></center></body> ");




            client.println("</html>");

            break;


            if (c == '\n')

            {
```

```
                currentLineIsBlank = true;
                }
              else if (c != '\r')
              {
              currentLineIsBlank = false;
              }
            }
          }
        }
      }
    delay(1);
    client.stop();
    Serial.println("client disonnected");
    digitalWrite(LED, LOW);
  }
}
```