

**INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**  
**GOIÁS**  
**Câmpus Jataí**

**Coordenação do Curso Superior de Tecnologia em  
Análise e Desenvolvimento de Sistemas**

**ROGÉRIO GONÇALVES ARAÚJO**

**MINI-ESTAÇÃO METEOROLÓGICA UTILIZANDO ARDUINO**

Jataí  
2018

# **MINI-ESTAÇÃO METEOROLÓGICA UTILIZANDO ARDUINO**

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Goiás - Campus Jataí como um dos pré-requisitos necessários para aprovação no Curso Superior Tecnológico em Análise e Desenvolvimento de Sistemas.

**Rogério Gonçalves Araújo**

**Orientador: Esp. Fabrício Vieira Campos**

Jataí, fevereiro de 2018.

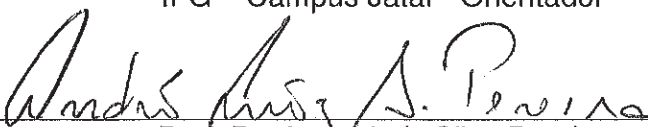
**Rogério Gonçalves Araújo**

## **MINI-ESTAÇÃO METEOROLÓGICA UTILIZANDO ARDUINO**

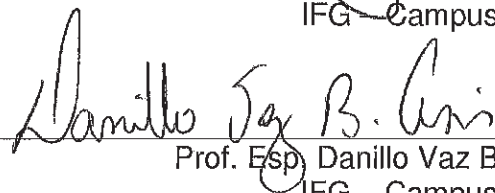
Trabalho de Conclusão de Curso, apresentado ao Instituto Federal de Goiás - Campus Jataí, como requisito parcial para aprovação no Curso Superior Tecnológico em Análise e Desenvolvimento de Sistemas. Sendo considerado **APROVADO** com nota final igual a 7,8, conferida pela Banca Examinadora formada pelos professores:



Prof. Esp. Fabrício Vieira Campos  
IFG – Campus Jataí - Orientador



Prof. Dr. André Luiz Silva Pereira  
IFG – Campus Jataí



Prof. Esp. Danilo Vaz Borges de Assis  
IFG – Campus Jataí

Jataí, 23 de fevereiro de 2018.

Dedico este trabalho a minha família que tanto me apoiou nesta minha jornada acadêmica e aos meus ilustres mestres que tive o prazer de conhecer ao longo de todo o curso.

## **Agradecimentos**

Agradeço meu orientador Fabrício Vieira Campos que tanto ajudou na criação deste projeto, compartilhando seus conhecimentos sobre o assunto abordado, bem como estimulando a obtenção de novos conhecimentos para poder chegar ao objetivo do presente trabalho.

“Lembre-se que as pessoas podem tirar tudo de você, menos o seu conhecimento.”

Albert Einstein

Rogério Gonçalves Araújo. **Mini-estação Meteorológica utilizando Arduino**: 2018. 71 folhas. Trabalho de Conclusão de Curso, Campus Jataí, IFGoias, Jataí, 2018.

## **Resumo**

Estudar e entender como as coisas funcionam sempre esteve presente no dia-a-dia da humanidade, entre estes itens é correto destacar o estudo sobre o clima e suas influências no campo e nas cidades. Observando a importância esse projeto se propôs a construir um protótipo de mini-estação meteorológica utilizando o Arduino, que é uma plataforma de prototipagem de baixo custo e de projeto aberto. Para sua realização foi necessário estudar o Arduino e suas diversas características técnicas que seriam necessárias para o desenvolvimento do produto final, também realizar testes de montagem e programação dos diversos sensores utilizados. Como produto final, obteve-se um protótipo em estágio inicial de uma mini-estação capaz de obter informações de temperatura, pressão, quantidade de chuva e altitude que são armazenados em um cartão SD. Desta forma foi possível observar que o Arduino é um dispositivo que atende as necessidades iniciais para o desenvolvimento da Mini-estação meteorológica proposta neste projeto e que demonstra potencial para novos projetos nessa área de monitoramento de informações utilizando sensores.

**Palavras-chave:** Arduino, Sensores, Meteorologia.

### **Dados Internacionais de Catalogação na Publicação na (CIP)**

	Araújo, Rogério Gonçalves.
ARA/min	Mini-estação meteorológica utilizando Arduino/ Rogério Gonçalves Araújo. -- Jataí: IFG/Coordenação dos cursos de Informática – Tecnologia em Análise e Desenvolvimento de Sistemas, 2018. 71 f.; il.  Orientador: Prof. Esp. Fabrício Vieira Campos. Bibliografias.  1. Arduino. 2. Sensores. 3. Meteorologia. I. Campos, Fabrício Vieira. II. IFG, Câmpus Jataí. III. Título
	CDD 005.1

Ficha catalográfica elaborada pela Seção Téc.: Aquisição e Tratamento da Informação.  
Bibliotecária – Rosy Cristina Oliveira Barbosa – CRB1/2380 – Campus Jataí. Cód. F034/18.



## Lista de Figuras

Figura 1 - Placa Arduino. ....	18
Figura 2 - Imagem Arduino Nano. ....	19
Figura 3 - Imagem Arduino Mega. ....	20
Figura 4 - Imagem Arduino Duemilanove. ....	20
Figura 5 - Alimentação da Placa Arduino. ....	21
Figura 6 - O Circuito Regulador Para Entrada Externa. ....	22
Figura 7 - Cirtuito de Proteção USB da Placa Arduino. ....	22
Figura 8 - Circuito de Seleção de Fonte do Arduino. ....	23
Figura 9 - Conectores de Alimentação R3 do Arduino. ....	24
Figura 10 - Conversor USB-serial com ATmega16u2. ....	25
Figura 11 - Circuito de Comunicação Serial. ....	26
Figura 12 - Pinagem ATmega328 Usado no Arduino UNO. ....	26
Figura 13 - Pinos de Entrada e Saída no Arduino UNO R3. ....	27
Figura 14 - Pinos ATmega328P. ....	28
Figura 15 - Parte Traseira do Arduino - Sem Isolação. ....	29
Figura 16 - Dimensão da Placa Arduino UNO. ....	30
Figura 17 - Resumo da Placa Arduino. ....	31
Figura 18 - Imagem da Raspberry Pi. ....	32
Figura 19 - Imagem da Beaglebone. ....	32
Figura 20 - Comparação Entre Placas de Prototipagem Com Arduino UNO. ....	33
Figura 21 - BMP180. ....	35
Figura 22 - DS18B20. ....	36
Figura 23 - Pluviômetro Báscula. ....	38
Figura 24 - DS3231. ....	39
Figura 25 - SDCard. ....	41
Figura 26 - Fritzing Tela Inicial. ....	42
Figura 27 - Modo Protoboard. ....	42
Figura 28 - Modo Esquemático. ....	43
Figura 29 - Modo PBC. ....	43
Figura 30 - Modo Código. ....	44
Figura 31 - Mini-estação Montada. ....	45
Figura 32 - Pluviômetro. ....	46
Figura 33 - IDE Arduino Detalhada. ....	47
Figura 34 - Listas de Arduinos. ....	48
Figura 35 - Acionando a Porta COM. ....	49
Figura 36 - Endereço Das Portas Serias de Um Sistema Linux. ....	50
Figura 37 - Selecionando o Exemplo Blink. ....	51
Figura 38 - Exemplo de Código. ....	51
Figura 39 - Tecla de Verificação de Código. ....	52
Figura 40 - Botão Upload. ....	53
Figura 41 - Exibição do Serial Monitor. ....	53
Figura 42 - Gráfico de Temperaturas internas, Externas e Pluviômetro. ....	56
Figura 43 - Gráfico de Pressão Atmosférica. ....	57
Figura 44 - Gráfico de Altitude. ....	57

## **Tabelas**

<b>Tabela 1 Ligação do Sensor BMP 180.....</b>	<b>35</b>
<b>Tabela 2 Ligação do Sensor DS18B20. ....</b>	<b>36</b>
<b>Tabela 3 Ligação do Sensor de pluviometria.....</b>	<b>37</b>
<b>Tabela 4 Ligação do Sensor DS3231 RTC.....</b>	<b>39</b>
<b>Tabela 5 Ligação do Sensor de SDCard.....</b>	<b>40</b>
<b>Tabela 6 Exemplo de Sados Coletados.....</b>	<b>59</b>

## Lista de Abreviaturas e Siglas

AVR	Microcontrolador RISC arquitetura 8 bit
CHIP	Microcircuito Integrado
CI	Circuito Integrado
CS	Chip Select
DIP28	Soquete de 28 pinos
D1	Diodo 1
EEPROM	Electrically-Erasable Programmable Read-Only Memory
FAT	File Allocation Table
ICSP	In - circuit serial programming
IDE	Integrated Development Environment
IOREF	I/O reference
I2C	Inter-Integrate Circuit
KB	Kilobyte
KBPS	Kilobyte Por Segundo
LED	Light Emiting Diode
L1	Bobina 1
MA	Mili Ampere
MISO	Master In Slave Out
MHZ	Mega Hertz
MOSI	Master Out Slave in
PWM	Pulse-Width Modulation
PWMs	Plural de PWM
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
RN3A	Resistor de 22 ohms
RN3D	Resistor de 22 ohms
RTC	Real Time Clock
RX	Recepção de dados
R3	Conector padrão do Arduino
SCL	Serial Clock
SDA	Serial Data
SDCARD	Secure Digital Card
SPI	Serial Peripheral Interface
TX	Transmissão de dados
USART	Universal Synchronous Asynchronous Receiver Transmitter
USB	Universal Serial Bus
U2	Circuito Integrado
VIN	Vehicle Identification Number
Z1	Varistor de proteção 1
Z2	Varistor de proteção 2

## Sumário

<b>1</b>	<b>INTRODUÇÃO .....</b>	<b>12</b>
1.1	OBJETIVO.....	13
1.2	JUSTIFICATIVA .....	13
1.3	METODOLOGIA.....	14
1.4	ORGANIZAÇÃO DO TRABALHO.....	15
<b>2</b>	<b>DESENVOLVIMENTO DO PROTÓTIPO .....</b>	<b>16</b>
2.1	ARDUINO .....	16
2.1.1	ARDUINO E SUAS ESPECIFICAÇÕES POR MODELO.....	19
2.1.2	PRINCIPAIS CARACTERÍSTICAS DO HARDWARE.....	21
2.2	CÉREBRO DO ARDUINO .....	26
2.3	PROGRAMAÇÃO DA PLACA ARDUINO UNO.....	29
2.3.1	RESUMO DA PLACA ARDUINO UNO .....	30
2.4	OUTROS DISPOSITIVOS DE PROTOTIPAGEM.....	31
2.4.1	RASPBERRY PI.....	31
2.4.2	BEAGLEBONE.....	32
2.4.3	INTEL GALILEO.....	32
2.5	SENSORES .....	34
2.5.1	BMP180.....	34
2.5.2	DS18B20 .....	36
2.5.3	PLUVIÔMETRO DE BÁSCULA .....	37
2.5.4	DS3231 .....	38
2.5.5	SDCARD .....	40
2.6	FRITZING: CRIAÇÃO DE DIAGRAMAS.....	41
2.7	TECNOLOGIAS UTILIZADAS.....	44
2.8	PROTÓTIPO MONTADO. ....	45
2.9	AMBIENTE DE DESENVOLVIMENTO INTEGRADO .....	46
<b>3</b>	<b>A MINI-ESTAÇÃO METEOROLÓGICA.....</b>	<b>55</b>
<b>4</b>	<b>TRABALHOS FUTUROS.....</b>	<b>58</b>
<b>5</b>	<b>EXEMPLOS DE DADOS COLETADOS .....</b>	<b>59</b>
<b>6</b>	<b>CONCLUSÃO .....</b>	<b>60</b>
<b>7</b>	<b>REFERÊNCIAS.....</b>	<b>61</b>
<b>8</b>	<b>ANEXOS .....</b>	<b>63</b>
8.1	CÓDIGO BMP 180.....	63
8.2	CÓDIGO DS18B20 .....	64
8.3	CÓDIGO DO PLUVIÔMETRO DE BÁSCULA.....	65
8.4	CÓDIGO DO REAL TIME CLOCK DS3231 .....	66
8.5	CÓDIGO PARA SDCARD: .....	68
8.6	CLASSE MAIN PARA CHAMAR TODOS OS SENSORES LIGADOS .....	70

## 1 Introdução

Há muito tempo o homem coleta e estuda os dados relacionados ao clima para entender e também prever o comportamento dos fenômenos da natureza ligados a ele, basta observar que em quase todos os telejornais, se não todos, existe um quadro destinado exclusivamente a mostrar a previsão de tempo indicando as condições de chuva, temperatura e umidade do ar.

Neste sentido Exposti (2013) diz que “As primeiras observações começaram na Grécia (VI a.C) e após no século V a.C quando Hipócrates escreveu o tratado das *“Águas, Ares e Lugares”*. Porém esta ciência tem seu início mesmo com Aristóteles”.

Com o crescimento das cidades as informações sobre a quantidade de chuva que poderá cair em determinado local são consideradas essenciais para ações rápidas de órgãos públicos como: Defesa Civil e Corpo de Bombeiros na prevenção de catástrofes e mortes de cidadãos. Crescimento esse muito vezes desordenados e sem planejamento, faz com que a cidade passe a ocupar cada vez mais regiões que antes eram consideradas impróprias para se morar, como beiras de rios, riachos e encostas de montanhas.

Já em outras regiões do país os mesmos dados referentes às chuvas são importantes devido à atividade econômica predominante na região, tais como a agricultura e pecuária. Informações essas que podem auxiliar os produtores a decidirem a data de plantio e ainda o tipo de cultura a ser plantada.

O Estado de Goiás é conhecido como “celeiro do Brasil” devido a atividade predominante de pecuária e agricultura, o que torna as informações destanatureza fator fundamental para as atividades fins.

Em todo o Brasil o órgão público mais conhecido para a coleta e divulgação de informações desse tipo é o Instituto Nacional de Meteorologia (INMET), que coleta e divulga informações meteorológicas baseado em informações coletadas em estações espalhadas em todo o país e também pela análise de imagens de satélites.

No entanto, diversos lugares ainda não possuem tal coleta de dados devido à inexistência de equipamentos capazes de coletá-los e enviá-los para posterior análise. Assim, o desenvolvimento de uma mini-estação meteorológica

utilizando Arduino, permitirá ter um produto de baixo custo podendo ser implantado em diversos lugares, contribuindo para o estudo desta área importante.

O clima é um recurso natural vital ao nosso bem-estar, saúde e prosperidade. As informações coletadas, gerenciadas e analisadas ajudam tomadores de decisão e usuários a planejar e adaptar suas atividades e projetos às condições esperadas. Desta maneira, decisões podem ser tomadas no planejamento, o que reduz riscos e aperfeiçoam os benefícios sócio-econômicos.(INMET, [1909], p.1).

## **1.1 Objetivo**

O objetivo deste projeto é criar um protótipo de uma mini-estação meteorológica de baixo custo, que permita o fácil acesso aos dados coletados sobre temperatura interna do dispositivo, pressão atmosférica, altitude, temperatura externa e medição de chuva em milímetros contendo data, mês, ano, horas, minutos e segundos.

## **1.2 Justificativa**

O desenvolvimento de uma mini-estação de meteorologia utilizando o Arduino passa a ser importante diante do cenário local, que é regido pela agricultura, ao permitir a criação de um protótipo que possa ser utilizado, principalmente, em pequenas e médias propriedades rurais permitindo aos produtores coletar informações sobre o clima, e consequentemente dar-lhes a oportunidade de fazer suas próprias análises de previsões de chuva e temperatura que auxiliem estes produtores na tomada de decisões em suas terras.

Na área, urbana por sua vez, este protótipo pode auxiliar as empresas locais a decidirem qual época se deve ou não começar reparos que dependam exclusivamente de um tempo estável para não comprometer todo o processo. Bem como auxiliar os órgãos responsáveis de determinada cidade a tomarem medidas protetivas contra as intempéries do clima, podendo minimizar possíveis percas materiais, perca de vidas ou até mesmo auxiliar nos vãos locais.

Podendo evitar assim tragédias como a ocorrida na Região Serrana do Rio de Janeiro no final do ano de 2010, a qual provocou 506 mortes, e foi considerada a maior tragédia climática da história país. O número de

vítimas ultrapassou o registrado em 1967, na cidade de Caraguatatuba, no litoral norte de São Paulo, onde naquela tragédia, tida até então como a maior do Brasil, 436 pessoas morreram. (PORTAL G1, 2011, p.5).

Com base em um estudo aprofundado do clima, tempo e geografia, os estudiosos conseguem explicar as causas dos repetidos desastres naturais que afligem não só o Rio de Janeiro mas também diversos locais em território nacional.

Excesso de precipitação, solo raso, vegetações rasteiras, mal planejamento do crescimento das cidades e relevo, se juntam numa perigosa mistura desencadeando deslizamentos de terra capaz de arrastar tudo por onde passam.

Sendo assim, observando a predominância de propriedades agrícolas na região do sudoeste de Goiás, pensar, e executar um projeto que vise auxiliar os produtores rurais na obtenção de dados meteorológicos passa a ter uma importância estratégica, ao tentar produzir um produto que é de baixo custo.

### **1.3 Metodologia**

Este trabalho pode ser considerado como uma pesquisa quanto aos objetivos do tipo exploratória, por buscar reunir informações sobre o desenvolvimento de protótipos utilizando a plataforma Arduino, neste sentido Moreira (2006) diz que pesquisa exploratória é “a que tem como principal finalidade desenvolver, esclarecer e modificar conceitos e idéias, com vistas à formulação de problemas mais precisos ou hipóteses pesquisáveis para estudos posteriores”.

Com o intuito de por em prática o que foi proposto, o projeto foi desenvolvido nas seguintes etapas:

- Pesquisa Bibliográfica – nesta fase foi feito a revisão da literatura com a finalidade de compreender sobre o clima, seus impactos na vida do ser humano, funcionalidades do Arduino, limitações e recursos do mesmo. Funcionamento dos sensores, cujo foco principal foi a descoberta do que são, como funcionam bem como configurá-los;
- Montagem da mini-estação meteorológica – nessa fase foram montados cinco sensores em uma mesma placa Arduino, e feito o teste de cada sensor de modo separado;

- Programação da plataforma: nessa fase foram criados os algoritmos capazes de ler todos os sensores e que a cada um segundo grava estes dados lidos em um cartão SD;
- Teste da mini-estação– nessa fase a mini-estação ficou em teste para verificar os dados coletados, verificar possíveis falhas bem como constatar as calibrações de cada sensor.

## 1.4 Organização do Trabalho

Este trabalho está organizado da seguinte forma:

- Capítulo 1: Descreve o objetivo do trabalho, sua metodologia bem como sua organização.
- Capítulo 2: Descrição do desenvolvimento do trabalho, explanando desde o Arduino, sensores usados, tecnologia aplicada no desenvolvimento e dados coletados.
- Capítulo 3: Descrição do ambiente de desenvolvimento do projeto.
- Capítulo 4: Conclusão do projeto, dificuldades e aferição de dados coletados.
- Capítulo 5: Listagem das referencias utilizadas na escrita desse trabalho.
- Capítulo 6: Transcrição dos códigos usados para cada sensor bem como a classe Main do projeto.



## 2 Desenvolvimento do Protótipo

### 2.1 Arduino

O Arduino é uma placa microcontroladora que contém circuitos de entrada/saída que pode ser conectada a um computador e programada via Ambiente de Desenvolvimento Integrado (do inglês: Integrated Development Environment - IDE) utilizando uma linguagem baseada em C/C++, sem a necessidade de equipamentos extras além de um cabo USB. Este projeto iniciou em 2005 na Itália contando com 5 membros: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mallis.

Uma das vantagens apresentadas por essa microcontroladora é o fato de que sua linguagem de programação se assemelha muito com a linguagem de programação C, isto faz com que aquelas pessoas que já conhecem essa linguagem de programação não necessitem aprender uma nova linguagem para poder programá-lo. Neste sentido Monk (2015) diz:

Uma concepção errônea a respeito do Arduino é que ele tem a sua própria linguagem de programação. Na realidade, ele é programado em uma linguagem denominada C. Essa linguagem já existe desde os primeiros dias da computação. O que o Arduino proporciona é um conjunto simples de comandos de fácil uso escrito em C que você pode usar em seus programas. Os mais tradicionais poderão dizer que o Arduino usa C++, a extensão orientada a objeto da linguagem C. Rigorosamente falando, isso é verdade, mas o Arduino dispõe de somente 1 ou 2 Kb de memória. Como resultado, as práticas encorajadas pela programação orientada a objetos não são uma boa ideia quando aplicadas ao Arduino. Na prática, fora alguns casos especiais, você está de fato programando em C. (MONK, 2015, p18).

Devido o fato de que o Arduino já é uma placa pronta, ou seja, traz consigo todos os componentes necessários para seu funcionamento, não sendo necessário grande conhecimento em eletrônica para poder utilizá-la. Esta característica permitiu que o mesmo fosse utilizado pelas mais diversas pessoas que saibam programação e que tinham pouco conhecimento em eletrônica.

Vale ressaltar também que o custo para a obtenção desta placa é considerado um valor baixo, sendo que no site do representante oficial no Brasil Robocore, o preço da mesma encontra-se R\$ 119,00, e quando montada comprando os seus componentes separadamente em lojas de eletrônica esse valor pode chegar a R\$ 50,00.

Outra vantagem do Arduino é que sua arquitetura é livre, ou seja, qualquer entusiasta ou desenvolvedor pode criar sua própria placa Arduino ou até mesmo fazer melhorias nos softwares de *bootloader*<sup>1</sup>, podendo assim melhorar o Arduino para melhor atender suas necessidades de desenvolvimento.

Esse fato fez também com que a quantidade de comunidades virtuais crescessem de maneira rápida, e da mesma forma que a arquitetura do projeto é livre, os códigos de programação para os seus diversos projetos também ficam disponíveis para consulta de maneira gratuita.

Pelo fato dessa placa possuir um circuito que permite a comunicação com um computador por via USB, este circuito tende a simular uma conexão como se fosse uma porta serial, onde por meio de uma IDE, pode-se criar a programação para resolver um problema específico como acender um Diodo Emissor de Luz(do inglês: Light Emitter Diode - LED), ligar um pequeno motor de corrente contínua, ler sensores, dentre outras funcionalidades.

Para as funções mais complexas, ou mais utilizadas e que não são encontradas disponíveis no Arduino o desenvolvedor dispõe de placas que podem desempenhar essas funções especiais. Estas placas são chamadas de *Shields*, são placas que normalmente se encaixam perfeitamente sobre o Arduino e disponibilizam funções das mais variadas, como rede cabeada, rede Wifi, portas para acionamentos de dispositivos elétricos entre outras.“As placas básicas são complementadas por placas acessórias, denominadas Shields, que podem ser encaixadas por cima da placa do Arduino.”(MONK, 2013, p1).

Por fim, é correto enfatizar a quantidade de áreas que tem utilizado o Arduino para resolver seus problemas, elas vão desde robótica, musica até impressoras em 3D“Os campos de atuação para o controle de sistemas são imensos, podendo ter aplicações na área de impressão 3D, robótica, engenharia de transportes, engenharia agrônômica, musical, moda e tantas outras.”(MULTILÓGICA-SHOP, 2009, p 14).

A placa representada pela figura 1 pode ser alimentada tanto pelo computador, utilizando a porta USB, quanto por uma fonte externa desde que não seja inferior a 7V contínuo nem ultrapasse a voltagem limite de 12V contínuo.

---

<sup>1</sup>Bootloader Carregador de arranque.

Figura 1-Placa Arduino.



Fonte: (THOMSEN 2014, p.2).

O Arduino é programável via comunicação serial pois o processador ATMEGA versão 168, 328 e 1280, tem um *bootloader* dispensando que o programador grave o binário na placa. O protocolo STK500 é o responsável pela comunicação.

Existem hoje, muitos modelos de Arduino, pois como fora dito anteriormente qualquer pessoa pode melhorar o modelo existente sem qualquer problema.

A caráter demonstrativo será apresentado os três modelos de Arduino mais usados atualmente.

Cada usuário irá determinar qual placa Arduino será usada em seu projeto, pois cada modelo possui características próprias para um tipo de projeto, como número de portas analógicas e digitais sendo as seguintes opções: o Arduino Uno e suas 14 portas digitais e 6 analógicas, passando por placas com maior poder de processamento; o Arduino Mega com microcontrolador ATmega2560 e 54 portas digitais; e o Arduino Due baseado em processador ARM de 32 bits e 512 Kbytes de memória. Sendo que cada Arduino contém um microcontrolador Atmel 328-P, controlador USB, circuito regulador de voltagem, plugue para alimentação externa e outros componentes que serão devidamente apresentados mais adiante.

O Arduino é uma pequena placa controladora contendo um plugue de conexão USB (universal serial bus\*) que permite a ligação com um computador. Além disso, contém diversos outros terminais que permitem a conexão com dispositivos externos, como motores, relés, sensores luminosos, diodos a laser, alto-falantes e outros. (MONK, 2013, p1).

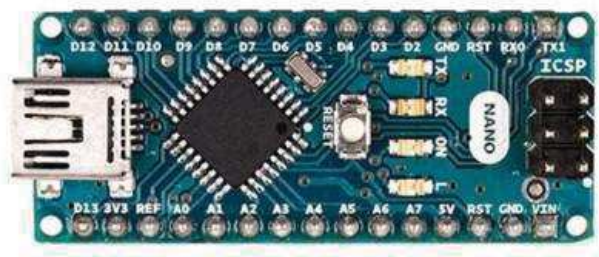
Seu baixo custo bem como a facilidade de ser programada por usar uma linguagem similar a C/C++ é um diferencial com relação a outras placas controladoras.

### 2.1.1 Arduino e suas especificações por modelo

#### 2.1.1.1 Arduino Nano:

A figura 2 representa o Arduino Nano.

*Figura 2-Imagem Arduino Nano.*



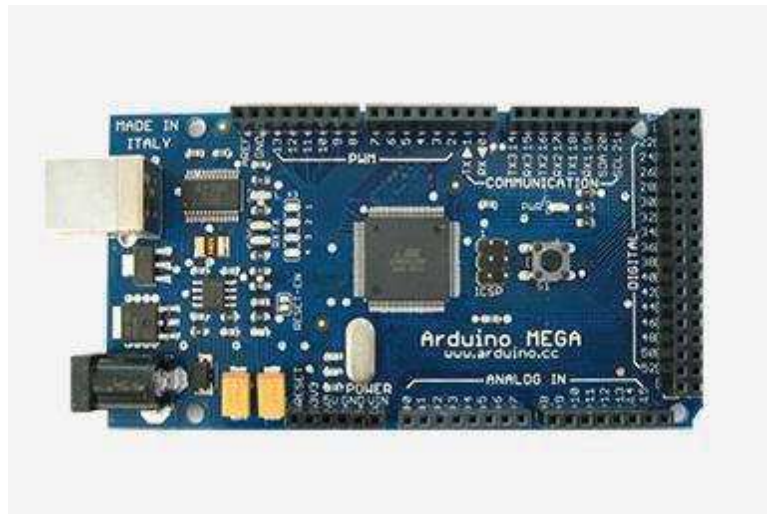
*Fonte: (STORE.ARDUINO.CC 2017, p.1).*

Possui praticamente as mesmas funcionalidades do Arduino Uno, porém com um tamanho reduzido, não possui conector para fonte externa e usa USB mini B ao invés do cabo USB convencional usado pelo Arduino Uno.

#### 2.1.1.2 Arduino Mega.

A figura 3 representa o Arduino Mega.

*Figura 3-Imagem Arduino Mega.*



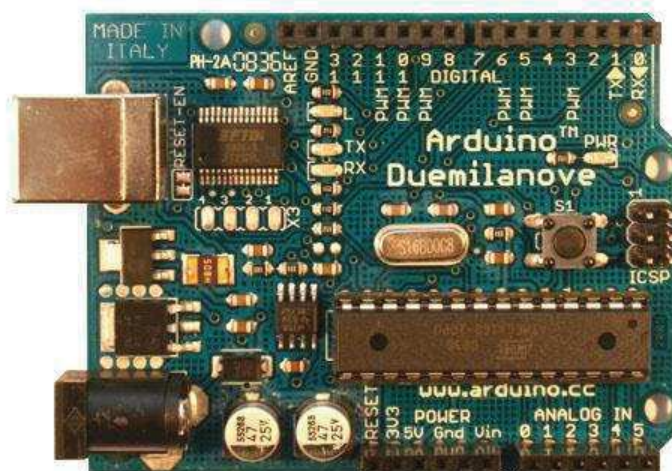
*Fonte:(STORE.ARDUINO.CC 2017, p.1).*

O Arduino Mega como o próprio nome diz, possui maior número de terminais e trabalha com um processador ATMEGA1280, 54 pinos de entrada e saída digitais, 16 pinos de entrada analógica e 4 UARTs(*Portas Seriais de Hardware*).

### **2.1.1.3 Arduino Duemilanove.**

A figura 4 representa a placa Arduino Duemilanove

*Figura 4-Imagem Arduino Duemilanove.*



*Fonte:(ARDUINO.CC 2009, p.1).*

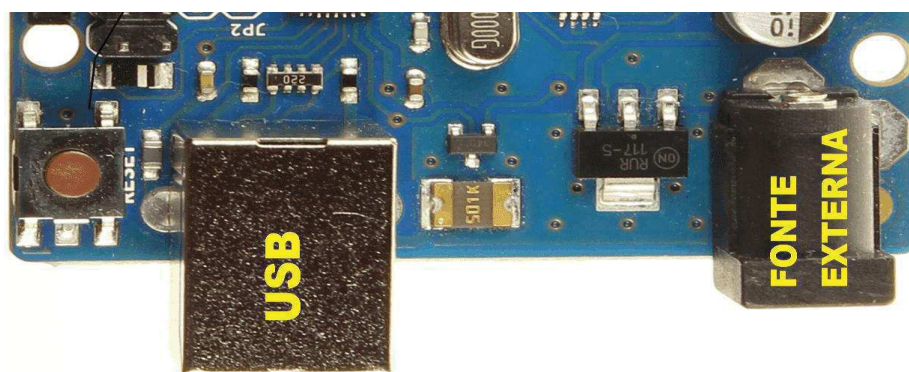


O Arduino Duemilanove é a versão do Arduino criada em 2009 por isto o nome duemilanove que é 2009 em italiano. Suas características são as mesmas do Arduino Uno tendo assim poucas diferenças entre os circuitos da placas.

### 2.1.2 Principais Características do Hardware

A figura 5 mostra que a alimentação da placa Arduino pode ser feita via USB ou por fonte externa.

*Figura 5 - Alimentação da Placa Arduino.*



*Fonte: (SOUZA 2013, p.2).*

Esta fonte externa é ligada por um conector *Jack* com centro positivo, sua variação de tensão não pode ser menor que 7V e nem superior a 12V, pois mesmo o circuito trabalhando com 5V se a placa receber alimentação inferior a 7V a linha de 5V da placa Arduino fica instável comprometendo assim o funcionamento do dispositivo.

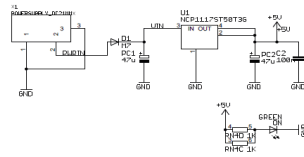
A figura 6 demonstra o esquema elétrico do bloco regulador de tensão da placa Arduino Uno.

Já por outro lado se a tensão for superior na 12V, haverá risco de aquecimento dos componentes bem como a avaria dos mesmos. Assim como na maioria dos circuitos eletrônicos, esta placa possui um CI responsável por regular a tensão de entrada evitando danos eventuais aos componentes, devido sobrecarga elétrica ou até mesmo ligações com polaridade invertida.

O CI responsável por esta tarefa é o NCP1117, *OnSemi* e é possível ver no esquema elétrico, conforme mostra a figura 7, deste bloco o uso do diodo D1

logo na entrada do *Jack* que protege o circuito caso seja ligado uma fonte com polaridade invertida.

*Figura 6- O Circuito Regulador Para Entrada Externa.*

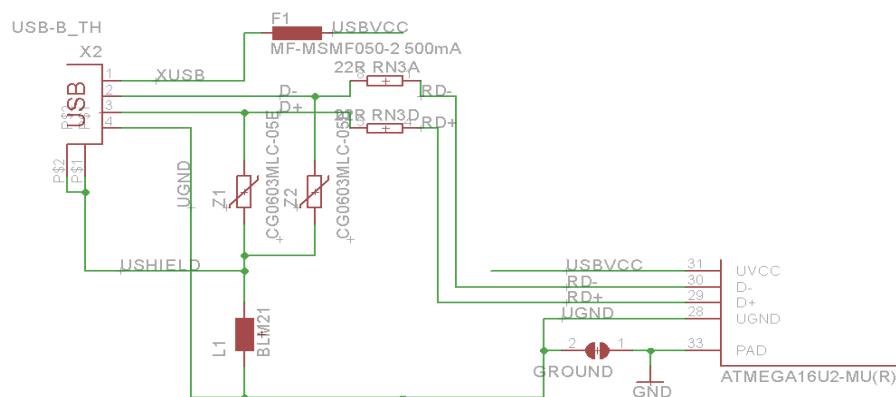


Fonte: (SOUZA 2013, p.3).

Estas medidas protetivas se fazem desnecessárias quando a alimentação do Arduino é feita pela porta USB, pois estas portas possuem uma tensão devidamente estabilizada.

A figura 7 descreve o circuito de proteção USB.

Figura 7 - Circuito de Proteção USB da Placa Arduino.



Fonte: (SOUZA 2013, p.3).

Mesmo se tratando de uma tensão regulada, seus desenvolvedores se preocuparam com a possibilidade de possível retorno de tensão podendo assim

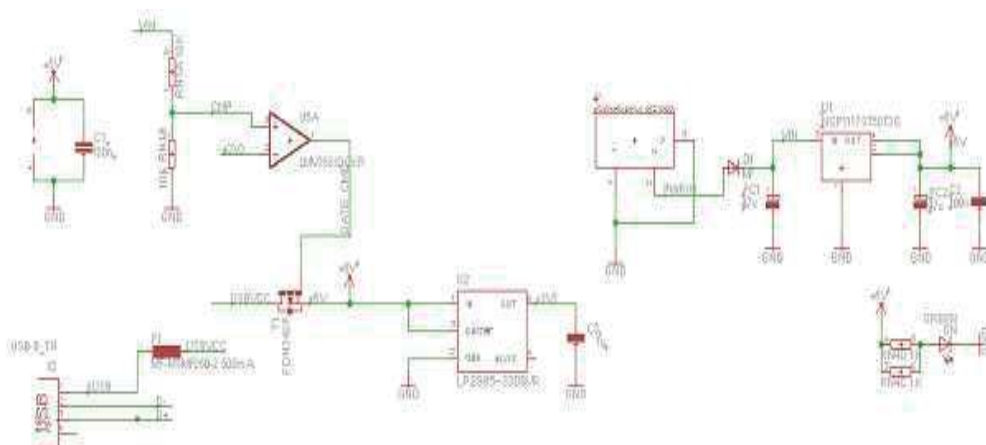
danificar a porta USB do computador. Os componentes Z1 e Z2 são varistores que funcionam abrindo o circuito ou entrando em curto circuito conduzindo a tensão diretamente para o terra ou GND.

Este mecanismo de proteção é falho por usar apenas os varistores sem um fusível de proteção em serie, pois os varistores possuem uma capacidade mais elevada de suportar sobrecargas, podendo assim permitir que mais componentes sejam avariados numa possível sobrecarga de tensão ou até mesmo num retorno de tensão.

Os resistores RN3A e RN3D que tem a função de limitador de tensão por esta razão sua resistência interna é de 22 Ohms, um fusível protetor da USB de 500mA e uma bobina L1 que irá filtrar quaisquer interferência que na eletrônica recebe o nome de ruído. Também vale lembrar que esta placa conta com um sistema de comutação, no qual, quando há a conexão de uma fonte externa e uma conexão via USB simultaneamente, a USB fica sendo apenas para comunicação com o computador e a tensão de 5V será feita apenas pela fonte externa.

A figura 8 mostra o circuito de seleção de fonte do Arduino.

Figura 8 - Circuito de Seleção de Fonte do Arduino.



Fonte: (SOUZA 2013, p.4).

A alimentação de 3.3V é feita pelo CI U2 o LP2985 que recebe a tensão de 5V e a converte em 3.3V, pois tanto o circuito interno do Arduino quanto algumas *Shields* necessitam desta tensão para seu funcionamento evitando também que a corrente ultrapasse os 50mA que é o limite pré estabelecido pelos



desenvolvedores do Arduino. A seguir a figura 9 irá mostrar os conectores responsáveis pelas alimentações de *shields* e demais componentes na placa Arduino Uno:

Figura9 - Conectores de Alimentação R3 do Arduino.



Fonte:(SOUZA 2013, p.4).

IOREF - Funciona como referência de tensão para as *Shields*. Este pino detecta a tensão de funcionamento de cada microcontrolador nele ligado e com base no que é detectado neste pino o circuito pode liberar a tensão correta para o funcionamento podendo variar de 3.3V a 5V.

RESET- Este pino é usado para resetar de forma externa o Arduino. Sua função é exatamente a mesma do botão de reset da placa do Arduino.

3.3V.- Alimentação de 3.3volts que serve para alimentar tantos as *Shields* quanto componentes externos com uma corrente máxima de 50mA.

5V.- Alimentação de 5volts que serve para alimentar tantos as *Shields* quanto componentes externos.

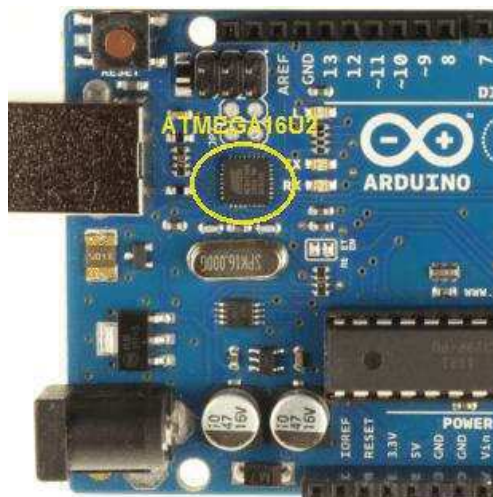
GND – Pino do aterramento ou negativo.

VIN – Este pino é usado para alimentar a placa por meio de fonte externa ou mesmo de uma bateria.

A grande desvantagem do Arduino é que a limitação da corrente que não pode ultrapassar 50mA.

A conexão USB se dá por meio de um CI ATMEL ATMEGA16U2 que possibilita a comunicação entre o Arduino e o computador, como mostra a figura 10.

Figura 10 - Conversor USB-serial com ATmega16u2.



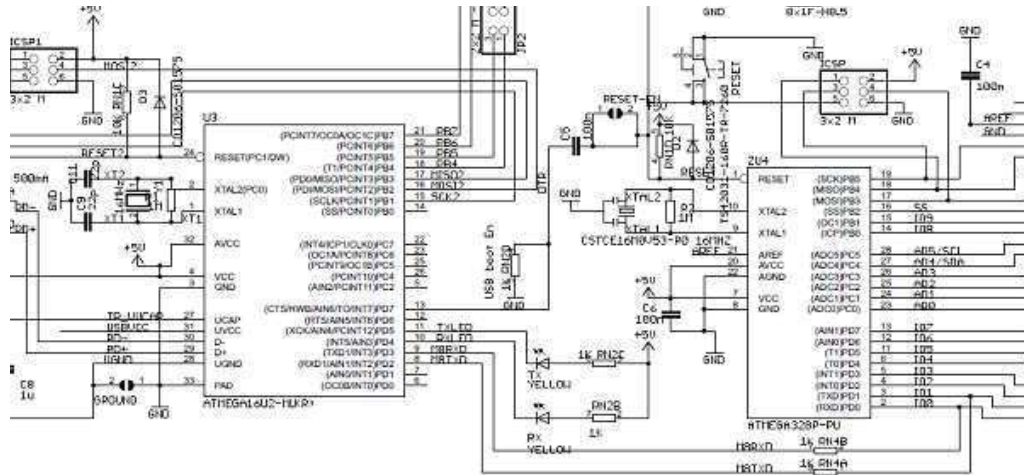
Fonte:(SOUZA 2013, p.6).

Esse circuito levará o código criado pelo usuário, quando acionado o botão Upload, do computador até a placa do Arduino. Ele possui um conector ICSP que pode ser usado para atualizar o firmware caso a ATMEL lance atualizações futuras. Os leds TX e RX são ligados a este circuito para indicar se está havendo ou não comunicação entre a IDE e o hardware.

O ATMEGA16U2 e o ATMEGA328 são conectados via canal serial de ambos os circuitos. Já o pino 13 do ATMEGA16U2 é ligado ao RESET do ATMEGA328 que faz com que o modo *bootloader* seja acessado de forma automática quando o botão Upload da IDE é acionada. Nas primeiras versões antes de acionar a tecla Upload a tecla RESET deveria ser acionada.

A figura 11 mostra o circuito de comunicação serial.

Figura 11 - Circuito de Comunicação Serial.

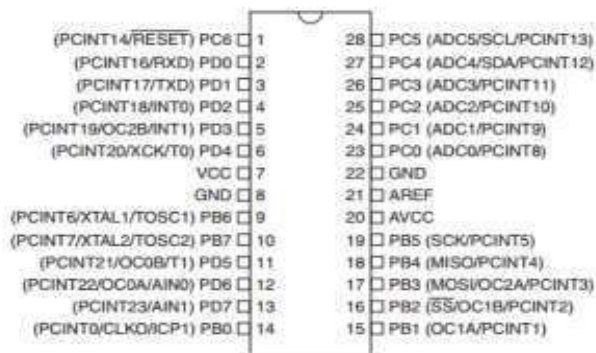


Fonte:(SOUZA 2013, p.7).

## 2.2 Cérebro do Arduino

O microcontrolador ATMEL ATMEGA328 é o cérebro do Arduino. Ele possui 8 bits de precisão da mesma linhagem dos AVR, arquitetura RISC e encapsulamento DIP28. Possui 32 KB de Flash sendo 512 Bytes reservado para o *bootloader*, 2 KB de Ram e 1 KB de EEPROM. Limite de operação de 12 MHz limitado pelo cristal externo ligado aos pinos 9 e 10 do microcontrolador. O circuito conta com 28 pinos, mas apenas 23 pinos podem ser usados como I/O como mostra a figura 12.

Figura 12 - Pinagem ATmega328 Usado no Arduino UNO.

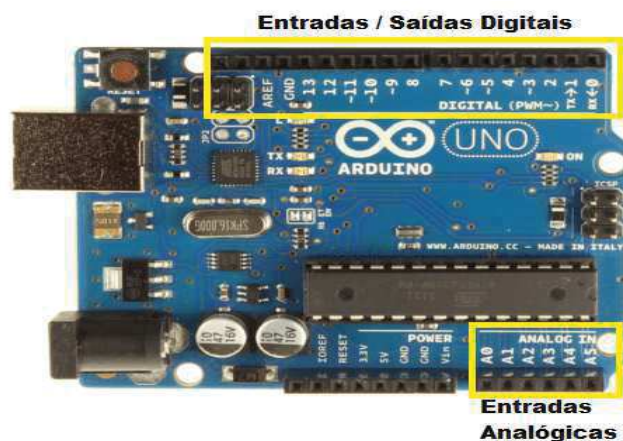


Fonte: (SOUZA 2013, p.7).

O ATMEGA328 pode funcionar com tensão até 1.8V, porém ele irá operar no máximo de 4MHz. Ele possui dois modos de economia de consumo o *Power-down mode* e o *Power-save mode* para economizar energia no modo de espera. Seus periféricos são USART de até 250Kbps, SPI até 5MHz e uma I2C que vai até 400KHz.

Existe também um comparador analógico interno ao CI e vários *timers*, bem como seis PWMs. Cada pino possui o limite de corrente de 40mA, mesmo assim a soma da corrente de todos os pinos não deve ultrapassar 200mA. Seu oscilador interno é de 32KHz usado por exemplo em situações de baixo consumo. O Arduino Uno tem pinos de entrada e saídas digitais, bem como pinos de entradas e saídas analógicas como mostra a imagem 13.

Figura 13 - Pinos de Entrada e Saída no Arduino Uno R3.



Fonte: (SOUZA 2013, p.8).

O dispositivo possui 14 pinos que podem ser usadas tanto para entrada digital quanto para saída digital. Os pinos funcionam com 5V. Seus pinos possuem de *pull-up* interno que podem ser acionados via *software*. Alguns pinos tem funções especiais:

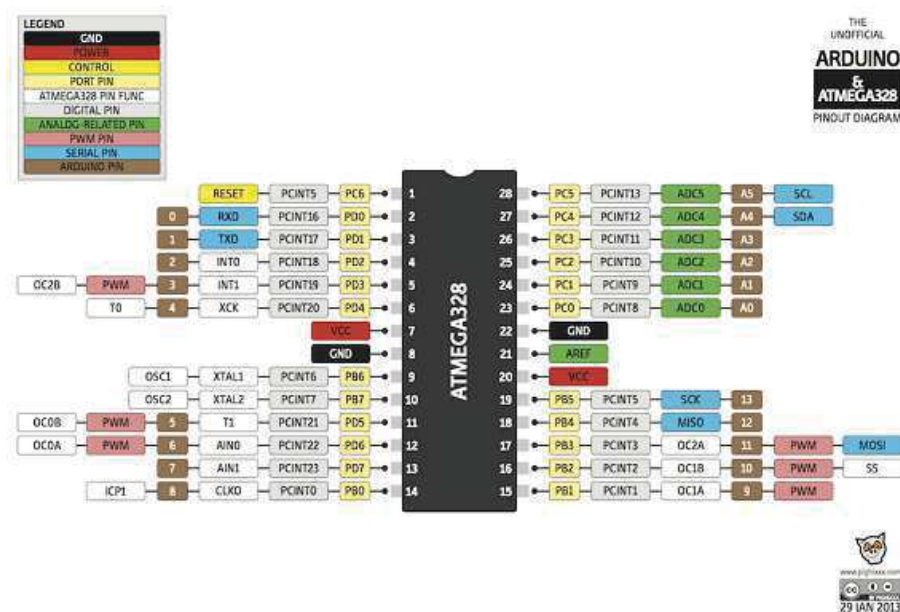
- PWM: 3,5,6,9,10 e 11 possuem o símbolo ~ significando que também são usada como PWM *out* com 8 bits de precisão via função *analogWrite()*;
- Comunicação serial: 0 e 1 RX e TX são usados para comunicação serial, lembrando que também são ligados ao microcontrolador que

comunica o USB ao computador;

- Interrupção externa: 2 e 3 por estes pinos é possível fazer uma interrupção externa usando a função *attachInterrupt()*;

Cada placa Arduino Uno possui 6 entradas analógicas com precisão de 10 bits. Os detalhes podem ser vistos na figura 14.

Figura 14 - Pinos ATmega328P.



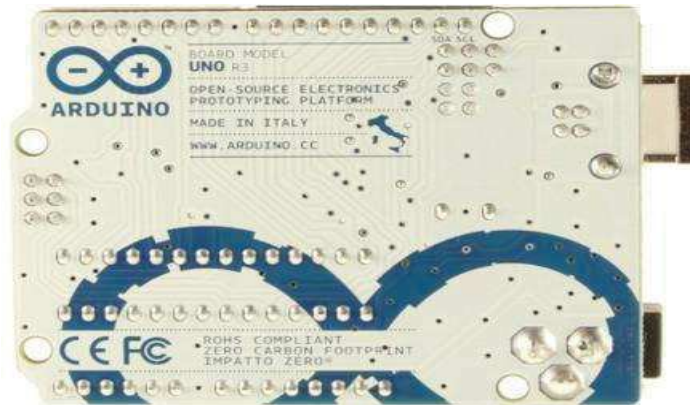
Fonte: (SOUZA 2013, p.9).

A manipulação deve ser cautelosa, pois não há nenhum componente protetivo entre os pinos e o microcontrolador, sendo assim qualquer erro pode danificar o circuito. Aconselha-se a sempre desligar a placa antes de conectar quaisquer periféricos, afim de, resguardar a integridade física do circuito.

Esta placa não conta com isolamento em sua parte traseira como mostra a figura 15, devendo ser fixada numa superfície isolante.



Figura 15 - Parte Traseira do Arduino - Sem Isolação.



Fonte:(SOUZA 2013, p.10).

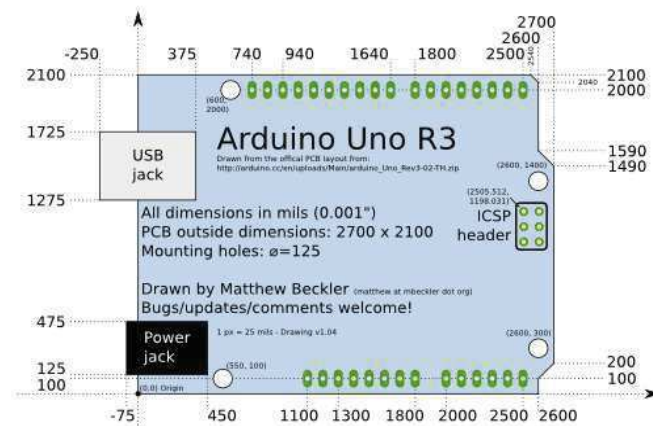
Esta placa é desprovida de botão *power*, todas as vezes que for cessar o funcionamento o cabo de alimentação deverá ser removido.

### 2.3 Programação da Placa Arduino Uno

O Arduino é programável via comunicação serial, pois o ATMEGA328 tem um *bootloader* dispensando que o programador grave o código binário na placa. O protocolo STK500 é o responsável pela comunicação. Sua IDE permite que a gravação do *software* desenvolvido pelo usuário seja feita diretamente no circuito ATMEGA 328 assim que a tecla Upload é clicada, facilitando seu uso.

Caso ocorra um corrompimento dos *bootloader* dos circuitos, tanto o ATMEGA328 quanto o ATMEGA16U2 podem ser programados via ICSP usando programas específicos da ATMEL. A placa possui dimensão muito pequena, cabendo na palma da mão, existem quatro furos para uma fixação, como mostra a figura 16.

Figura 16 - Dimensão da Placa Arduino UNO.



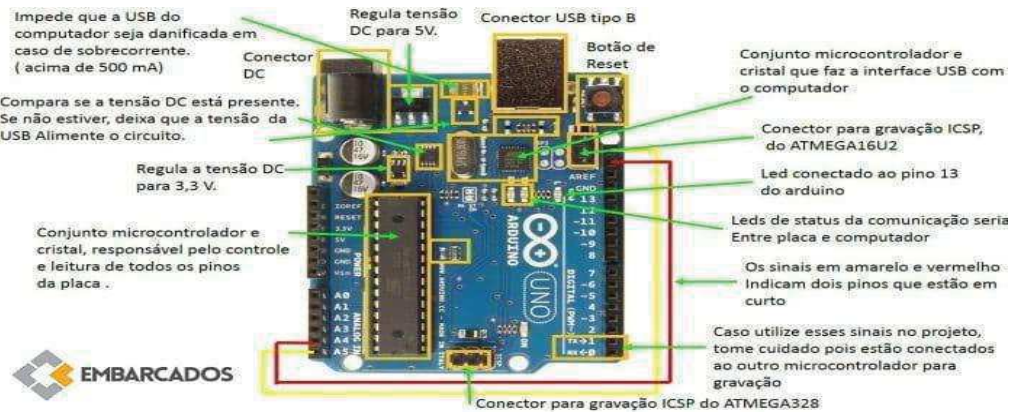
Fonte: (GOMBA 2011, p.1).

### 2.3.1 Resumo da placa Arduino Uno

O Arduino é uma ótima ferramenta para iniciantes, tem um baixo custo, uma liberdade tanto para melhoria de hardware como do *software* interno pode ser aperfeiçoado por seus usuários. Porém, por ser uma ferramenta simples o Arduino não conta com um sistema de *debug*, não tem como colocar *breakpoint* ou mesmo parar o *firmware* em tempo real para verificar endereços de memória nem mesmo usar *try catch*.

Para uma melhor compreensão a fim de detalhar cada componente existente no Arduino Uno a figura 17 irá mostrar as especificações de cada componente.

Figura 17 - Resumo da Placa Arduino.



Fonte:(SOUZA 2013, p. 11).

Cada componente está devidamente identificado com suas funcionalidades, sendo que alguns estão agrupados em blocos definidos por suas funções.

## 2.4 Outros Dispositivos de Prototipagem

Existem inúmeras placas de prototipagens hoje além do Arduino, como: Banana Pi, intel Galileo, PIC40, CORIDIUM ARMmit PRO e muitos outros, porém será abordado apenas as duas mais usadas apenas em caráter de comparação.

### 2.4.1 Raspberry Pi

Esta placa em especial é literalmente um mini computador de baixo custo, podendo ser usada também para projetos de prototipagem. Nesse caso para cada aplicação deverá ser reservado um cartão SD de no mínimo 8GB para que todos os recursos da placa sejam utilizados com maior aproveitamento. Ela possui saída de áudio e vídeo bem como conexões USB para teclado e mouse. Existe distribuições linux específicas para funcionar com este computador portátil, a figura 18 descreve a placa *Raspberry Pi*.



*Figura 18 - Imagem da Raspberry Pi.*



*Fonte:(UPTON 2015, p.1).*

Com esta placa pode-se criar um servidor web, um arcade, um mini computador, robôs, automação residencial podendo ligar ou apagar luzes remotamente, dentre outras aplicações.

#### **2.4.2 Beaglebone**

Esta placa em especial veio para suprir as lacunas encontradas no Arduino como, por exemplo, a necessidade de placas auxiliares para comunicação Ethernet, cartão USB, saída de áudio. Embora sua arquitetura seja aparentemente mais robusta, sua aplicação assim como a do raspberry pi requer configurações para funcionamento como prototipagem, a figura 19 representa a placa Beaglebone.

*Figura 19 - Imagem da Beaglebone.*



*Fonte:(GROENEVELD 2013, p.2).*

#### **2.4.3 Intel Galileo**

A placa Intel Galileo traz uma fusão dos recursos do Arduino com

todos os recursos e rapidez que os processadores Intel trazem. Dessa forma essa placa pode trabalhar com um Sistema Operacional Linux ou Windows, bem como interagindo com as *shields* do Arduino.

A placa Intel Galileo é a primeira placa com pinagem padrão Arduino que possui um processador de arquitetura Intel. Anunciada na Maker Faire Roma do ano passado, essa placa de desenvolvimento traz como principal atração o processador Intel Quark SoC X1000, o primeiro produto dessa linha, desenhada para aplicações que demandam baixo consumo e alta performance, além da já conhecida facilidade de prototipagem da plataforma Arduino.

É bem fácil de usar e foi pensada para iniciantes que queiram fazer projetos rápidos, que podem até ter um nível de complexidade mais alto. Chega ao mercado com um preço bem atraente, o que permite que hobbyistas e curiosos possam também se arriscar a trabalhar com ela em seus pequenos projetos ou descobertas. Essa placa é um projeto completamente open-source, com todo o esquema eletrônico e os detalhes da placa disponíveis para download. Provavelmente o primeiro hardware completamente open-source da Intel. (LIMA, 2014, p.1).

Na figura 20 descreve os recursos existentes em cada dispositivo.

Figura 20 - Comparação Entre Placas de Prototipagem Com Arduino UNO.

Nome	Arduino Uno	Raspberry Pi	BeagleBone
Modelo	R3	Modelo B	Black
Preço	R\$75,00	R\$250,00*	R\$230,00*
Tamanho	2,95"x2,10"	3,37"x2,125"	3,4"x2,1"
Processador	Atmega328	ARM11	ARM Cortex-A8
Velocidade de Clock	16MHz	700Mhz	700MHz
RAM	2KB	256MB	256MB
Flash	32KB	Cartão SD	4GB (microSD)
EEPROM	1KB	-	-
Tensão de alimentação	7V a 12V	5V	5V
Corrente mínima	42mA	700mA	170mA
pinos GPIO	14	8	65
Entradas analógicas	6	-	7
PWM	6	-	8
I2C	2	1	2
SPI	1	1	1
UART	1	1	5
Interface de Desenvolvimento	Arduino	IDLE, Scratch, Squeak/Linux, Python	Scratch, Squeak, Linux, Python
Ethernet	Apenas com uso de shield	10/100	10/100
USB	-	2 USB 2.0	1 USB 2.0
Saída de vídeo	Pode-se montar	HDMI, RCA	HDMI
Saída de áudio	-	Jack	HDMI

Fonte:(NATANAEL 2013, p.1).

Com esses recursos a disposição esta placa se mostra atrativa tanto para iniciantes quanto para desenvolvedores experientes.

## 2.5 Sensores

Sensores são transdutores de energia convertendo energia física, térmica, luz e outras formas de energia, em pulsos ou energia elétrica. Um exemplo clássico são os termômetros de mercúrio, que transformam a energia térmica em energia física dilatando o mercúrio em um invólucro devidamente mensurado para mostrar a temperatura ambiente ou corporal.

Cada tipo de sensor utilizado possui componentes específicos que transformam as variantes climáticas em questão nos pulsos elétricos necessários para leitura mediante um *software* específico.

### 2.5.1 BMP180

Este sensor, como mostra a figura 21, é um medidor de pressão atmosférica, temperatura e altitude. Com base na pressão atmosférica este dispositivo pode calcular a altitude com muita precisão. Sua biblioteca para aplicação em Arduino é a mesma utilizada pelo modelo anterior o BMP080 sendo assim não necessita de nenhuma modificação na mesma.

Este sensor usa a função *bmp.begin()* para inicializar seu funcionamento e traz uma pré-configuração interna com um escalonamento de processamento, onde primeiro ele inicializa o termômetro, pressão atmosférica para só depois calcular a altitude. Nos testes realizados foi possível verificar que este sensor tende a conflitar com o pluviômetro fazendo com que os dados computados não sejam exatos, sendo esta uma das dificuldades encontradas durante o desenvolvimento do projeto.

A tabela 1 descreve sua ligação com o Arduino.

Tabela 1 Ligação do Sensor BMP 180.

Porta ou cabo do sensor	Porta Arduino
SCL	pin A5
DAS	pin A4
GND	GND
VIN	3.3V

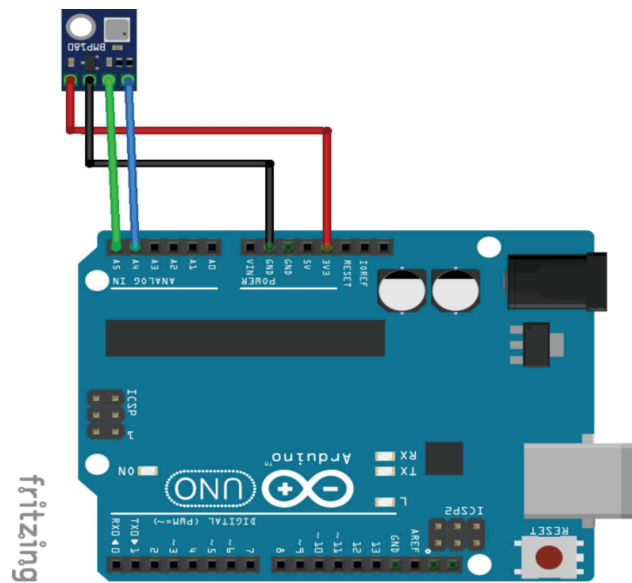
Fonte: Autoria própria.

Especificações:

- De 30.000 e 110.000 Pa para pressão atmosférica
- De -45° a 85° para temperatura
- Precisão de  $\pm 0.3^{\circ}\text{C}$
- De -1000 a 9000 metros de altitude

Possui 8 bits de precisão.

Figura 21 - BMP180.



Fonte: Autoria própria.

Infelizmente mesmo o Arduino possuindo uma linguagem C++, ele não possui *try catch*, sendo assim não foi possível sanar esta falha.

2.5.2 DS18B20

O DS18B20 é um termômetro blindado como é ilustrado pela figura 22. Seu circuito interno permite medir temperaturas que vão de -55°C a +125°C com uma precisão de ±0.5°C entre -10°C a +85°C. Sua pinagem geralmente é Vermelha, branco e preto, sendo que na maioria das vezes vermelho é a alimentação de 5V, preto é o GND e branco o sinal de saída.No Arduino, para que possa ser reconhecido a ligação deve ter o acréscimo de um resistor de 4.7k entre o pino vermelho e o branco.

A tabela 2 descrevesua ligação com o Arduino.

Tabela 2 Ligação do Sensor DS18B20.

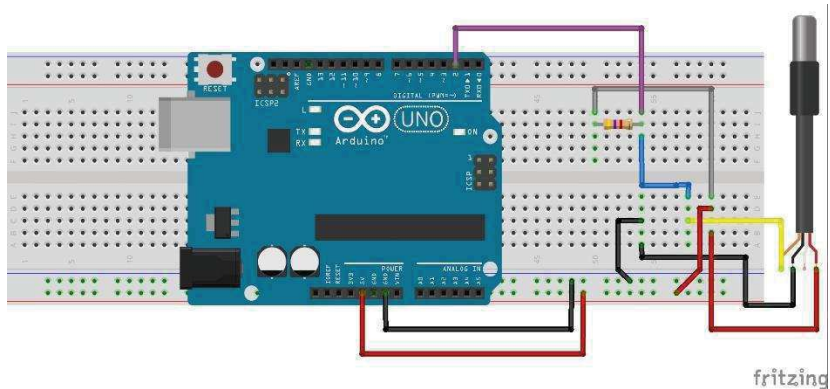
Porta ou cabo do sensor	Porta Arduino
Vermelho	pino 5V
Preto	GND
Amarelo	pino 2 da conexão digital
Amarelo	Resistor de 4.7k entre 5V e pino 2

Fonte: Autoria própria.

Especificações:

- Tensão de operação: 5 V.
- Medidas: mede temperaturas de -55°C a +125°C
- Precisão de ±0.5°C entre -10°C a +85°C.

Figura 22 - DS18B20.



Fonte: Autoria própria.

Deve-se também sempre lembrar de verificar as especificações junto ao fornecedor do produto, pois as pinagens podem mudar de um fabricante para outro. Com sua função OneWire ds (X); onde x será o número da porta digital que será usado, pode-se selecionar uma porta específica para melhor aproveitamento do Arduino.

Embora os sensores BMP180 e DS3231 possuam termômetros eles não podem ficar expostos ao tempo por possuírem circuitos elétricos que ao entrarem em contato com a chuva irão se danificar.

Já a blindagem do DS18B20 o torna ideal para aferições de temperaturas em ambiente externo.

### 2.5.3 Pluviômetro de Bâscula

Este modelo de pluviômetro possui um ímã preso ao centro lateral do balsa, fazendo com que a cada vez que um lado enche com a água da chuva sua balança ou balsa penda para baixo esvaziando o recinto, ao mesmo tempo que sua outra cavidade se enche de água fazendo que o balsa fique num movimento rítmico subindo e descendo enquanto na água da chuva estiver caindo em sua balança, o sensor é descrito pela figura 23.

A tabela 3 descreve sua ligação com o Arduino.

*Tabela 3 Ligação do Sensor de Pluviometria.*

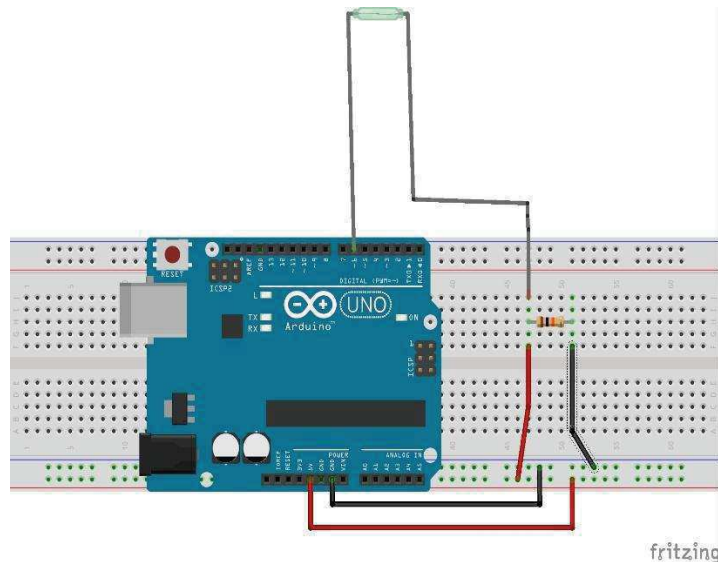
Porta ou cabo do sensor	Porta Arduino
Pino 1	pino 5V
Pino 2	pino 8 da conexão digital
Pino 2	Resistor de 10k entre GND e o pino 8

*Fonte: Autoria própria.*

Especificações:

- Tensão de operação: 5 V.
- Computa chuva mediante balança balsa com ímã.
- Cada passagem do ímã pelo sensor equivale a 0.25mm.

Figura 23 - Pluviômetro Báscula



Fonte: Autoria Própria.

Com um sensor de pulso de ímã fixado próximo à báscula no meio da balança é possível conseguir captar cada oscilação da báscula computando assim com base em quantos milímetros são necessários para manter seu movimento. A cada vez que o ímã passa perto do sensor tanto na ida como na sua volta o código irá ler como um movimento e converter este contador em milímetros de chuva.

#### 2.5.4 DS3231

O DS3231 é um Relógio de Tempo Real (do inglês: Real Time Clock – RTC) de baixo custo e ótima eficiência em comparação aos RTCs disponíveis atualmente no mercado. A figura 24 demonstra a placa DS3231.

A tabela 4 descreve sua ligação com o Arduino.



Tabela 4 Ligação do Sensor DS3231 RTC

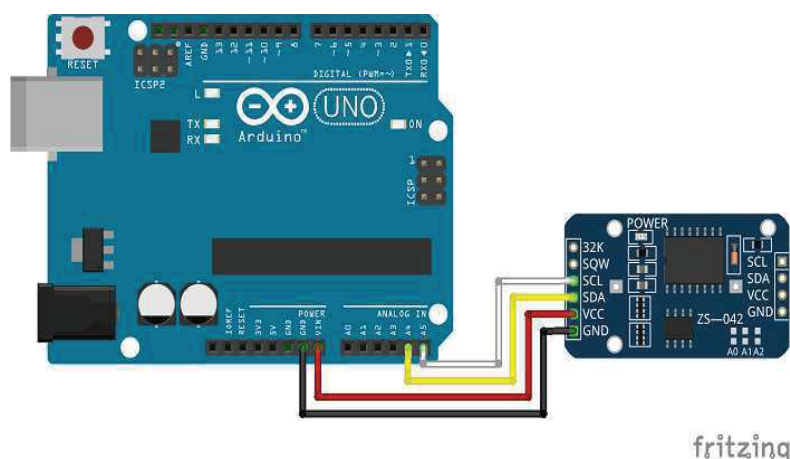
Porta ou cabo do sensor	Porta do Arduino
SLC	pino A5
DAS	Pino A4
VCC	pino VIN
GND	pino GND

Fonte: Autoria própria.

Especificações:

- Tensão de operação: 3V3 a 5 V.
- Computa segundos, minutos, horas, dias da semana, dias do mês, meses e anos(de 2000 a 2099).
- Sensor de temperatura com  $\pm 3^{\circ}\text{C}$  de exatidão.
- Chip de memória AT24C32(Capacidade de 32K que podem ser usadas como RAM estendida do microcontrolador).
- Interface I2C.
- Circuito de detecção de falha de energia.
- Consumo de 500nA no modo bateria.
- Faixa de temperatura de  $0^{\circ}$  a  $40^{\circ}\text{C}$ .
- Dimensões da placa 38x22x14mm.

Figura 24 - DS3231.



Fonte:(MARTINS 2017, p.3).



Ele consegue exibir meses com menos de 31 dias, formato de horas 12 ou 24 bem como correções de ano bissexto. Este tipo de dispositivo existe dentro de muitos aparelhos, como relógios digitais, despertadores, micro-ondas, entre outros, que mesmo após serem desligados, podem manter a contagem correta do tempo até serem ligados novamente.

### 2.5.5 SDCard

Para que seja possível gravar os dados em um cartão de memória é preciso o uso de uma placa SDCard.

A figura 25 demonstra a placa *SDCard*.

A tabela 5 descreve sua ligação com o Arduino.

*Tabela 5 Ligação do Sensor de SDCard.*

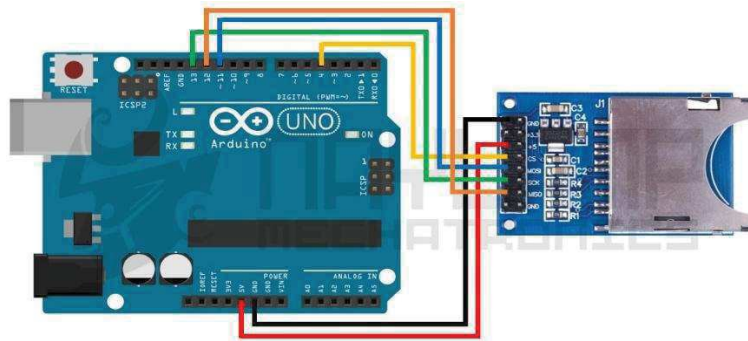
Porta ou cabo do sensor	Porta Arduino
<b>+5</b>	pino 5V
<b>GND</b>	GND
<b>CS</b>	pino 4 da conexão digital
<b>MOSI</b>	pino 11 da conexão digital
<b>SCK</b>	pino 13 da conexão digital
<b>MISO</b>	pino 12 da conexão digital

*Fonte: Autoria própria.*

Especificações:

- Tensão de operação de 3V3 a 5V.
- Interface SPI: MOSI, SCK, MISO e CS.
- Dimensões: 5.1 x 3.1cm.
- Formatação: FAT16 ou FAT32.

Figura 25 - SDCard.



Fonte:(NAYLAMPMECHATRONICS 2016, p.4).

Esta placa em especial permite a leitura ou gravação de um cartão SD. Faz-se necessário usar esta placa para armazenar os dados coletados em um cartão SD que deve estar formatado em FAT16 ou FAT32.

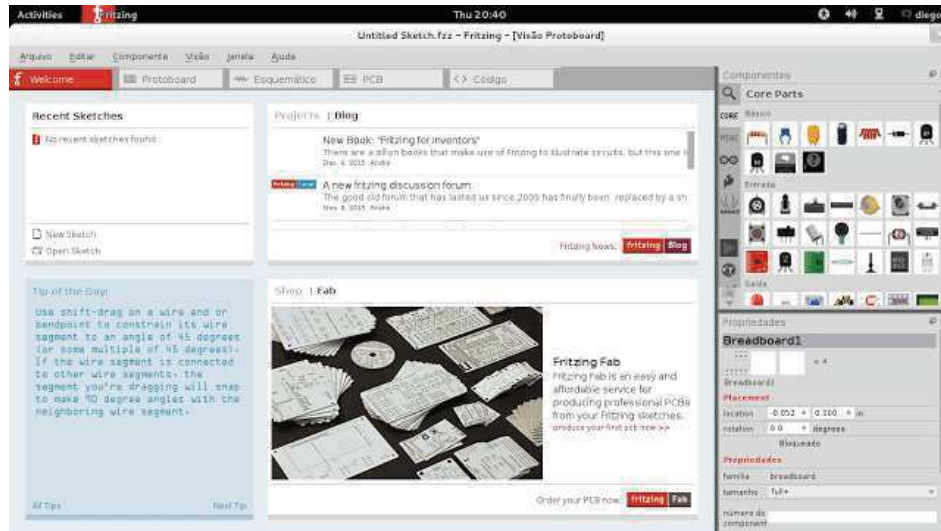
## 2.6 Fritzing: criação de diagramas

O Fritzing permite fazer esquemas eletrônicos, gerar imagens de projetos e até mesmo fazer simulações de funcionamento. Ele foi utilizado nesse projeto com o objetivo de criar os diagramas de ligação do Arduino com os sensores, de maneira a facilitar a visualização do leitor.

Ao Abrir a aplicação o usuário irá se deparar com uma interface intuitiva como demonstra a figura 26, bem como com uma grande gama de componentes elétrico eletrônico que vão desde CIs até *Shields*. Uma vez montado o circuito na aplicação Fritzing, é possível exportar este circuito em forma de PNG,PDF ou JPEG.

Fritzing é uma iniciativa de hardware open-source multiplataforma, que foi desenvolvida nos laboratórios da Universidade Aplicada de Postdam, na Alemanha. O software é destinado a criação de esquemas e diagramas eletrônicos, prototipagem e layout de placas de circuito impresso(PCB) usado com placas Arduino, Raspberry pi e BeagleBone.(SEMPREUPDATE, 2017, p.2).

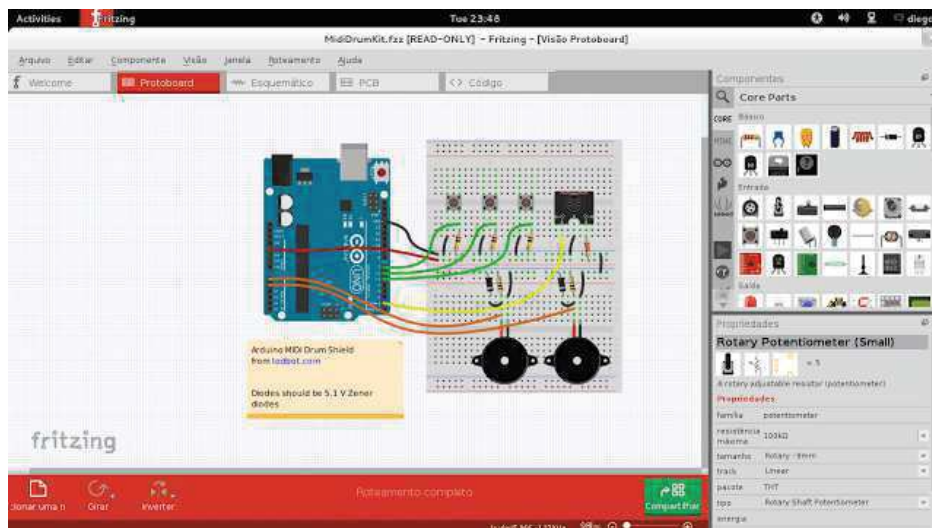
Figura 26 - Fritzing Tela Inicial.



Fonte:(SEMPREUPDATE 2017, p.2).

Quando se usa o Fritzing no modo protoboard como a figura 27 demonstra, pode-se facilmente selecionar as peças desejadas que são exibidas no lado direito da aplicação.

Figura 27 - Modo Protoboard.

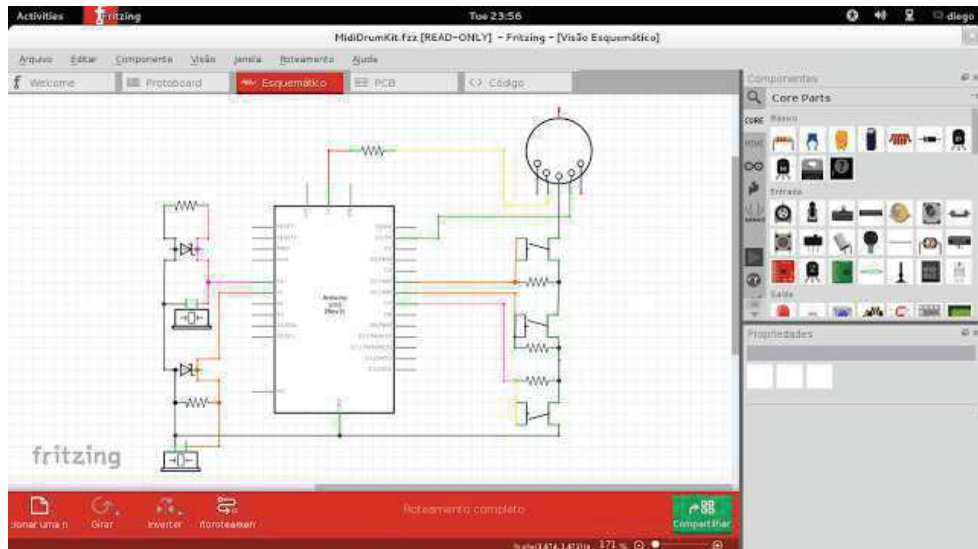


Fonte:(Sempreupdate 2017, p.3).

Quando se trabalha em modo esquemático ele converte as imagens

dos componentes em seus símbolos eletrônicos como demonstra a figura 28.

*Figura 28 - Modo Esquemático*



Fonte:(SEMPREUPDATE2017, p.4)

No modo PBC como demonstra a figura 29, o Fritzing mostra como a placa criada ficará. Desta forma a montagem de circuito impresso se dá com menor chance de erro na sua confecção.

*Figura 29 - Modo PBC*

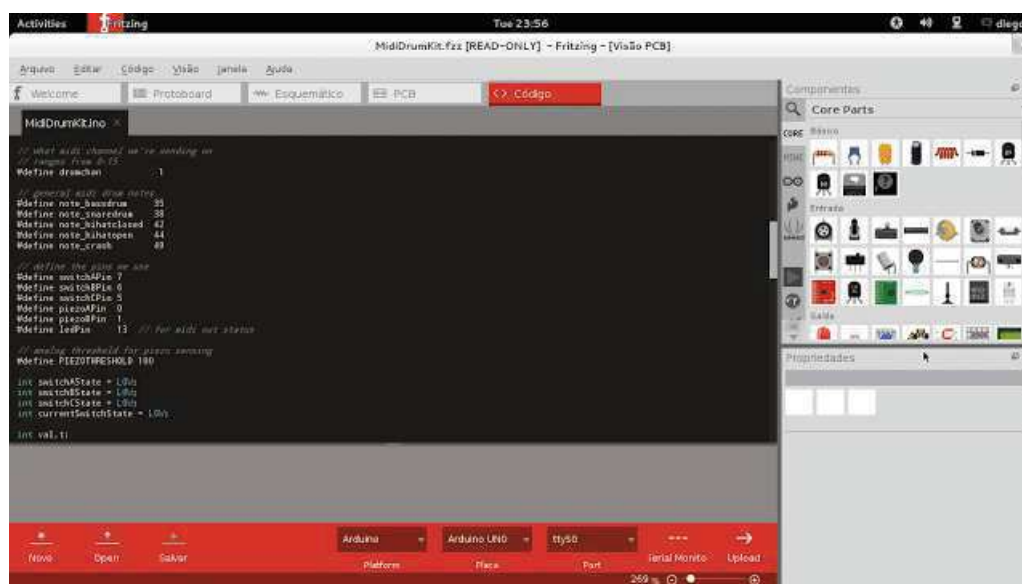


Fonte:(SEMPREUPDATE 2017, p.4)

O Fritzing conta também com o modo código como demonstra a figura 30, que tende a substituir a IDE da placa que se quer trabalhar.No caso do

Arduino, será capaz de poder criar um código, alterá-lo e gravá-lo diretamente na placa Arduino pro meio de um cabo USB sem precisar de sua IDE original.

Figura 30 - Modo Código



Fonte: (SEMPREUPDATE 2017, p.5)

Dessa forma, caso o usuário descida usar apenas o Fritzing, poderá criar todas as imagens necessárias bem como programar o Arduino pelo próprio Fritzing.

## 2.7 Tecnologias Utilizadas

Para este projeto foi utilizado uma plataforma já conhecida Arduino, que possui uma linguagem de programação similar a C++. Foram integrados sensores específicos para cada evento climático e devidamente calibrados para uma obtenção de dados para que demonstrasse o funcionamento, sendo que um desempenho mais exato seja algo para uma segunda fase do projeto ou para algo já a nível comercial.

A princípio este projeto teve uma fonte de alimentação retificada, usando a própria rede elétrica, porém futuramente poderá contar com uma placa solar, podendo assim ser instalada na zona rural sem maiores problemas. Todas as

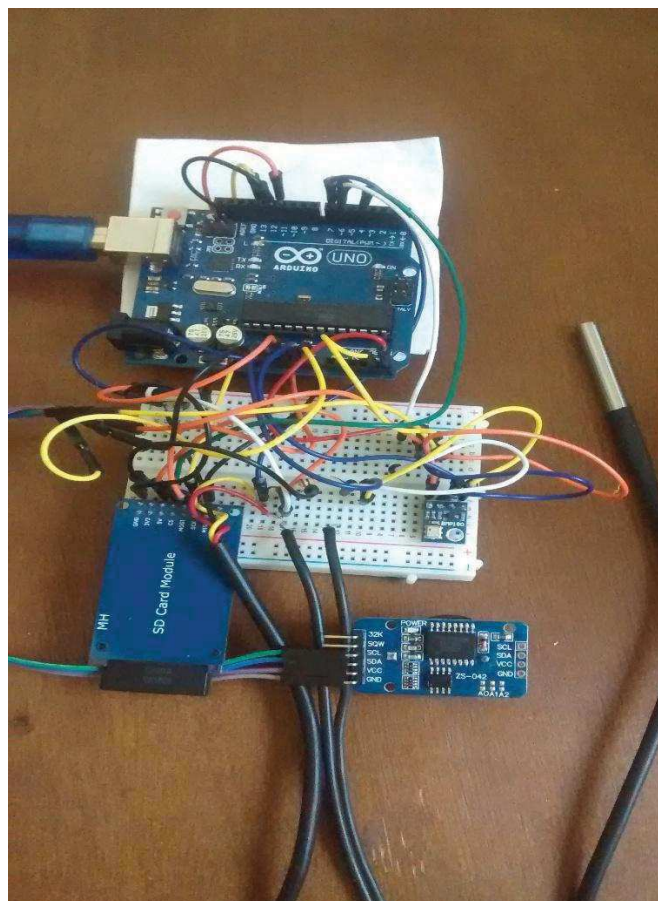


informações coletadas foram gravadas em cartão de memória SD, periodicamente com data e hora para que elas ficassem não só armazenadas mas também organizadas.

## 2.8 Protótipo Montado.

Segue a imagem da mini-estação meteorológica montada demonstrada pela figura 31.

*Figura 31 - Mini-estação Montada*



*Fonte: Autoria Própria*

A figura 32 demonstra pluviômetro devidamente instalado na parte externa para captação do nível de chuva.

*Figura 32 - Pluviômetro.*



*Fonte: Autoria Própria.*

Para coletar os dados referentes a chuva, o pluviômetro foi instalado onde o acesso a chuva não teve obstáculos como: telhados, marquises, toldos e calhas.

## **2.9 Ambiente de Desenvolvimento Integrado**

O IDE como demonstra a figura 33, é uma ferramenta de desenvolvimento do próprio Arduino. Esta ferramenta é o elo entre o desenvolvedor e a placa Arduino, fazendo com que o software desenvolvido possa ser gravado no CI ATMEL 328 via cabo USB.



Figura 33 - IDE Arduino Detalhada.

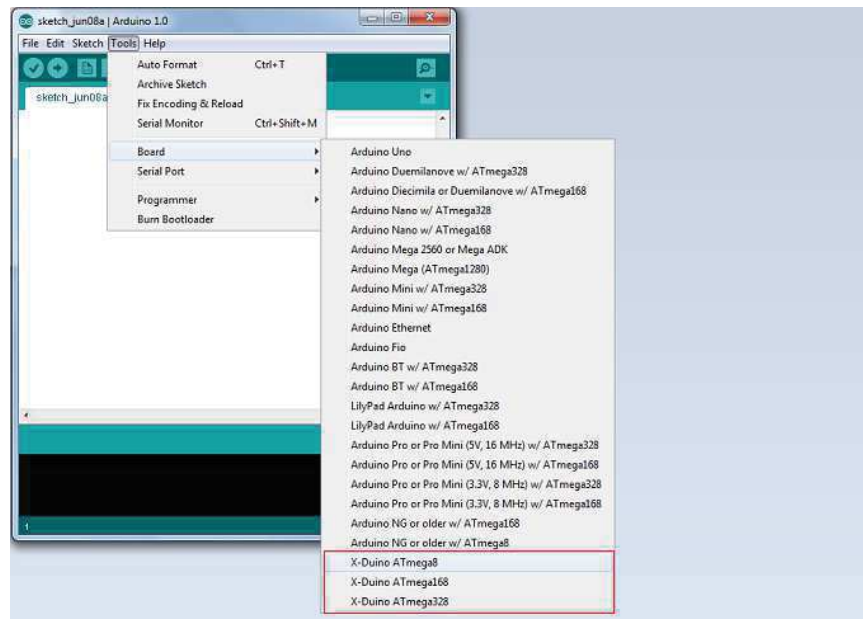


Fonte: (BASCONELLO FILHO 2016, p.3).

- Menu principal – No menu principal tem todas as funções da IDE, onde o usuário irá poder desde configurar a IDE até abrir exemplos já existentes na sua biblioteca.
- Barra de botões – Nesta barra o usuário irá encontrar botões responsáveis por salvar programa, abrir programa, novo programa, Upload que grava o programa na placa Arduino, verificação a fim de testar se o código escrito pelo programador e o serial monitor que exibe a aplicação que está sendo executada no interior do ATMEL 328 do Arduino.
- Console de programação – Este console tem como função mostrar os erros existentes no programa escrito na IDE.
- Barra de estado – Esta barra exibe qual Arduino está configurado e em uso.

Esta IDE também requer uma configuração para que funcione de forma correta como demonstra a figura 34. Na aba Tools é possível ver uma gama de Arduinos presentes. Atente sempre para o número correto do CI ATMEL usado e selecione a opção correta. No caso do Arduino UNO o ATMEL é o 328.

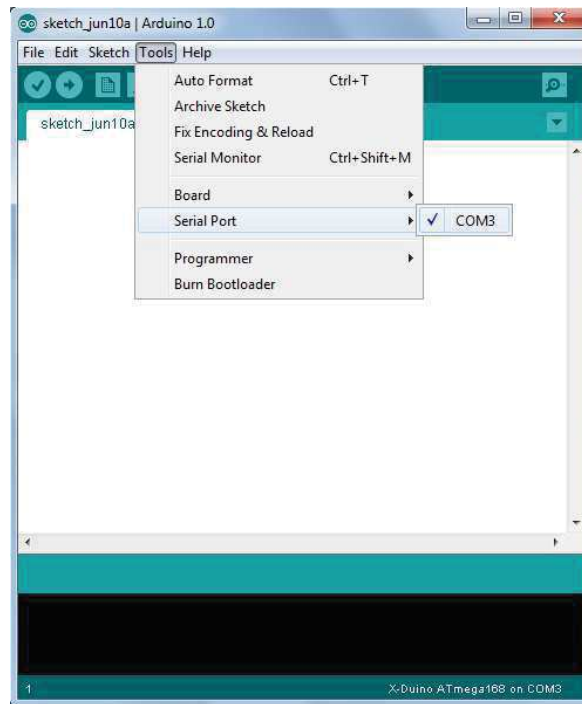
Figura 34 - Listas de Arduinos.



Fonte: (Basconello Filho 2016, p.4).

O circuito responsável pela comunicação entre a IDE e a placa do Arduino simula uma comunicação serial como demonstra a figura 35, emulando uma porta COM que deve ser marcada para que a IDE possa funcionar.

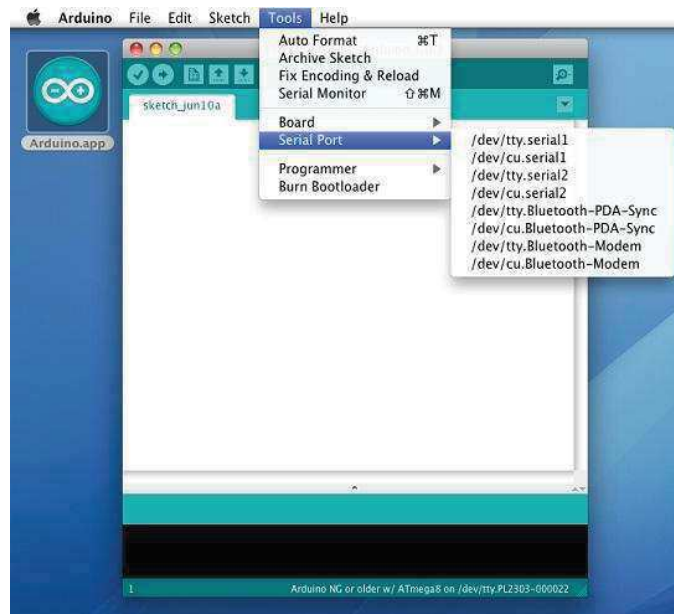
*Figura 35 - Acionando a Porta COM.*



*Fonte: (BASCONELLO FILHO2016, p.5).*

Cada sistema operacional trata de forma distinta o endereço das portas seriais como demonstra a figura 36, sendo assim deve-se ter sempre atenção quando as configurações forem feitas nas distribuições Linux.

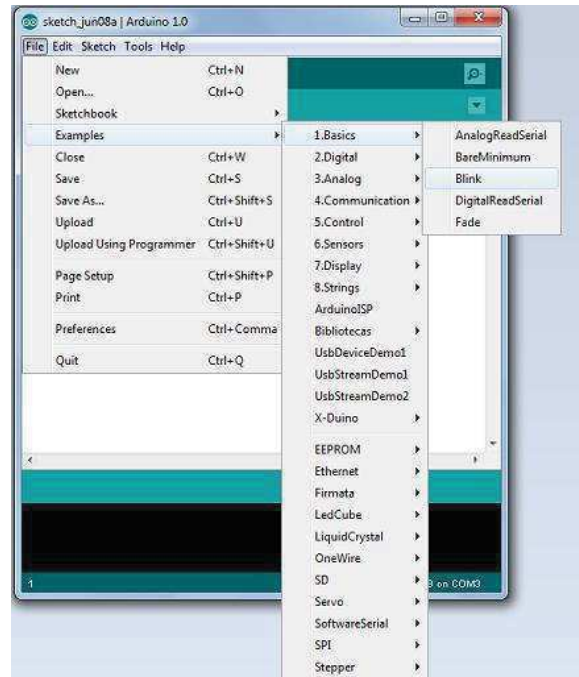
Figura 36 - Endereço Das Portas Serias de Um Sistema Linux.



Fonte: (BASCONELLO FILHO 2016, p.6).

Esta aplicação é muito atrativa por possuir uma biblioteca interna com alguns exemplos de programação, o que facilita para o programador na hora de desenvolver um programa para Arduino. Em um dos exemplos existe um recurso chamado *blink* como demonstra a figura 37, que aciona o LED existente na placa Arduino, comprovando assim a conexão entre IDE e a placa Arduino.

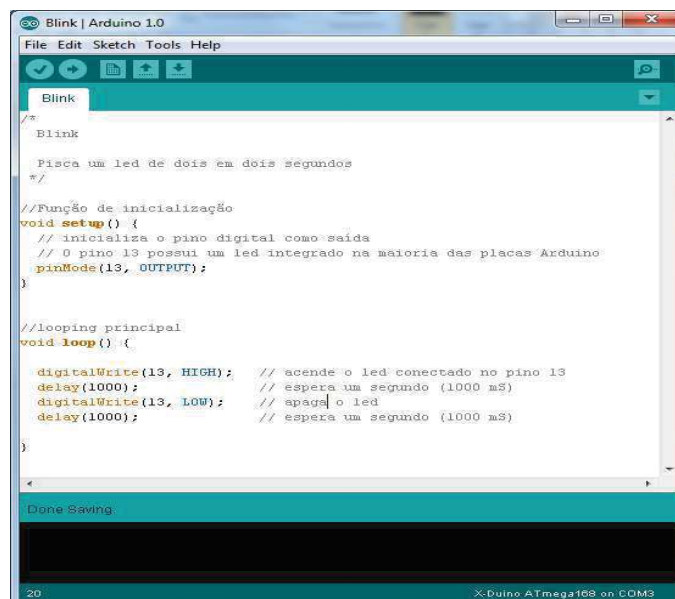
Figura 37 - Selecionando o Exemplo Blink.



Fonte: (BASCONELLO FILHO 2016b, p.2.).

A programação é dividida em duas partes como a figura 38 demonstra: O setup que contém comandos de inicialização, e o *loop* que será responsável pela repetição da execução do código enquanto a placa do Arduino estiver sendo alimentada.

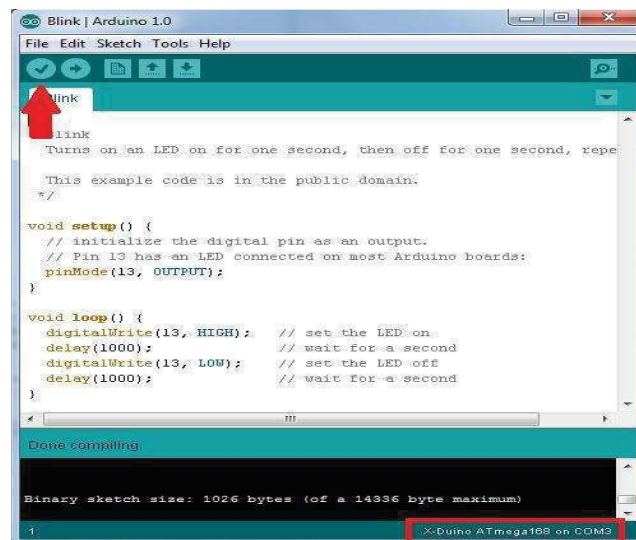
Figura 38 - Exemplo de Código.



Fonte: (BASCONELLO FILHO 2016b, p.3).

O botão V como demonstra a figura 39, é usado para verificar se o código está correto antes de gravá-lo no ATMEEL-328 do Arduino. Lembrando que existindo algum erro a programação criada não será gravada no Arduino.

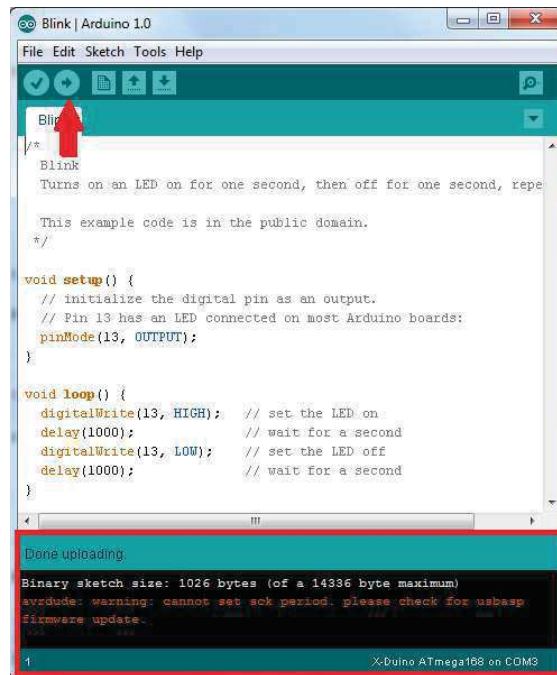
*Figura 39 - Tecla de Verificação de Código.*



*Fonte: (BASCONELLO FILHO2016, p.8).*

O botão *Upload* como demonstra a figura 40, gravará o programa desenvolvido pelo programador direto no Arduino. Caso exista algum erro será exibido no console.

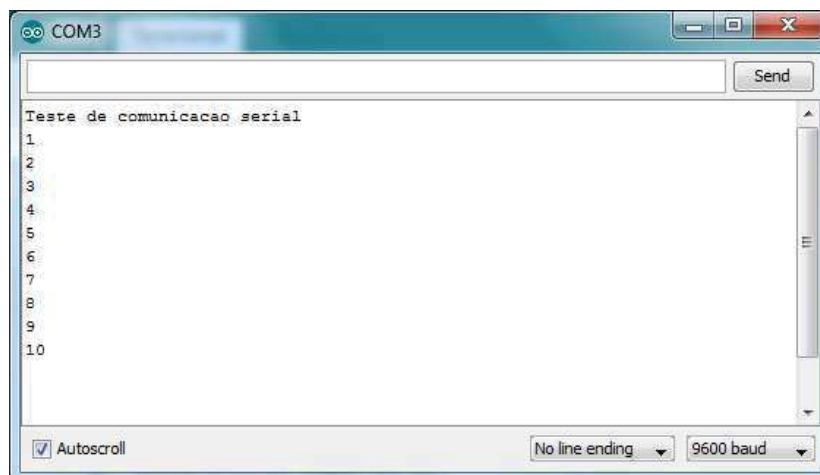
Figura 40 - Botão Upload.



Fonte: (BASCONELLO FILHO 2016b, p.10).

Após a gravação do programa desenvolvido na placa Arduino, é possível contar com a função serial monitor como demonstra a figura 41, para exibir o funcionamento do programa.

Figura 41 - Exibição do Serial Monitor.



Fonte: (BASCONELLO FILHO 2016c, p.3).



Como é possível observar na figura acima, o serial monitor retorna em tempo real a execução do programa gravado na placa Arduino.

### 3 A Mini-estação Meteorológica.

A mini-estação meteorológica criada tem por objetivo colher dados sobre, temperatura externa, temperatura interna, pressão atmosférica, altitude, milímetros de chuva, data, hora, por meio dos sensores acima citados e gravar estas informações em um cartão de memória SDCard.

Este projeto surgiu através da sugestão do professor e orientador deste trabalho de criar um protótipo usando as tecnologias acima citada, afim de, conhecer tanto o desempenho quanto as limitações e conflitos do Arduino com relação ao uso dos sensores.

Cada sensor vem com sua programação definida pelo fabricante e com retorno de valores do tipo numérico.

Para poder fazer a chamada dos valores lidos foi necessário converter estes valores para *CHAR* e assim fazer a chamada destes valores por meio de uma classe *MAIN*.

Com esta mini-estação foi possível conseguir as leituras de temperatura externa, temperatura interna, pressão atmosférica, altitude, data e hora, sem quaisquer alterações no funcionamento dos sensores. Já o pluviômetro por sua vez teve conflito com o BMP180 devido ao laço de repetição interno contido dentro do CI do sensor, isto faz com que haja uma lacuna de aproximadamente três a quatro segundos sem que a informação do pluviômetro seja lida.

Por o Arduino Uno não possuir *try catch* não foi possível sanar este conflito, sendo assim o problema ainda persiste.

Como o protótipo é de caráter exploratório, ou seja, visa conhecer e explorar as limitações do Arduino, os dados gerados, são linhas de texto, cada linha, contendo apenas os valores lidos, separados por ponto e vírgula para facilitar a importação em outras plataformas como EXCEL, por exemplo, onde o usuário irá poder gerar gráficos para uma melhor compreensão destas informações.

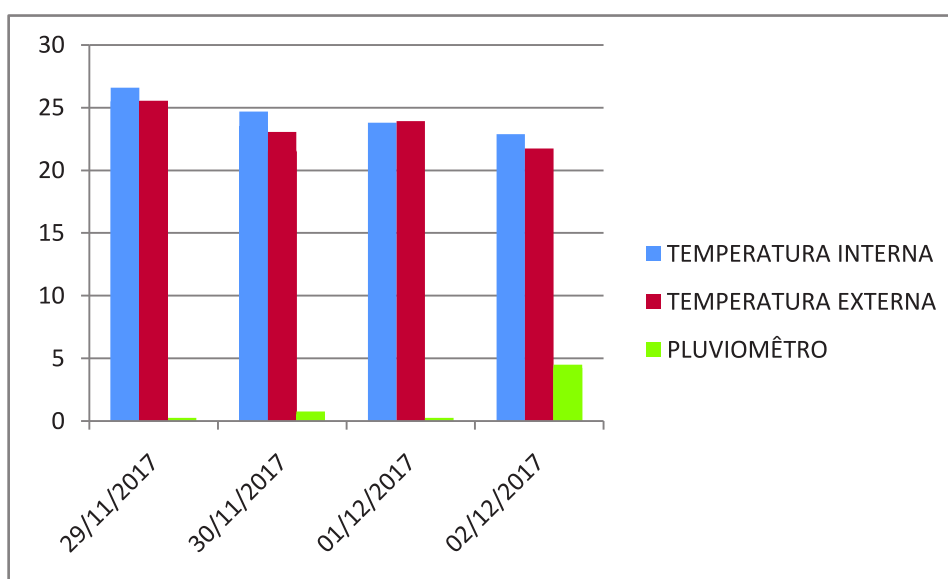
Como as informações são gravadas a cada um segundo no cartão de memória, basta o remover da placa SDCard, para coletar os dados gravados.

Lembrando sempre que quando for recolocar o cartão de memória de volta na placa SDCard o usuário terá que resetar a placa Arduino para que a gravação possa voltar a ser feita. Caso isto não seja feito a mini-estação continuará lendo as informações porém não as gravará no cartão de memória.

Veja na figura 42 o gráfico mostra os dados colhidos de data, temperatura interna, pressão atmosférica, altitude e os milímetros de chuva.

Usando os dados coletados pela mini-estação meteorológica foram gerados os gráficos por meio do EXCEL.

*Figura 42-Gráfico de Temperaturas internas, Externas e Pluviômetro.*

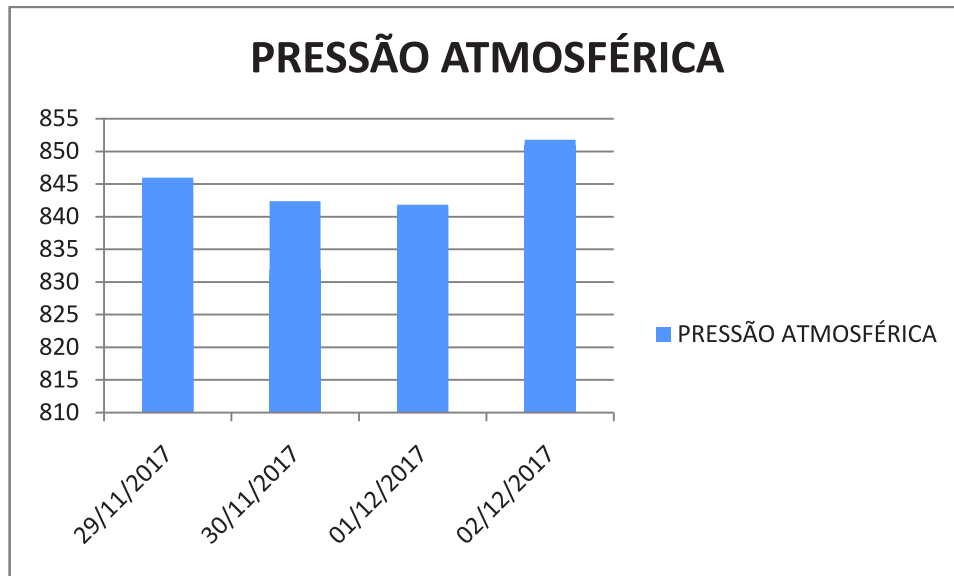


*Fonte: Autoria própria.*

Este gráfico mostra as variações dos valores lidos entre termômetro interno, externo e o pluviômetro.

A figura 43 mostra o gráfico gerado contendo data e pressão atmosférica.

Figura 43-Gráfico de Pressão Atmosférica.

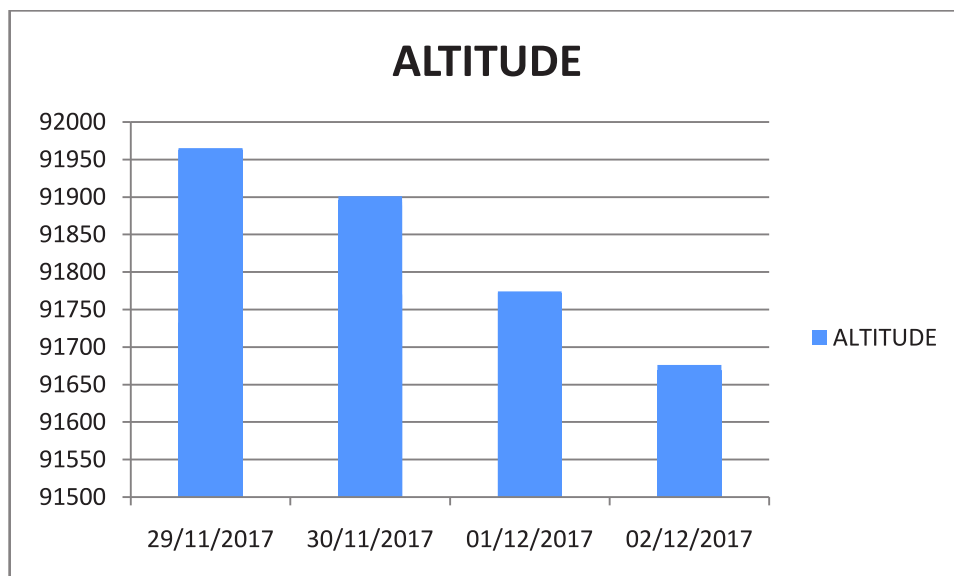


Fonte: Autoria própria.

No gráfico acima se pode notar a variação da pressão atmosférica lida pelo sensor.

A figura 44 mostra o gráfico referente à altitude.

Figura 44-Gráfico de Altitude.



Fonte: Autoria própria.

A figura acima mostra a variação da altitude lida pelo sensor.

Os gráficos acima foram gerados com os dados coletados que consta no capítulo 5, por meio do EXCEL.

## **4 Trabalhos Futuros**

Para trabalhos futuros deve se resolver o conflito entre o pluviômetro e o BMP 180 por meio de uma pesquisa mais detalhada acerca deste conflito ou até mesmo testes com outros Arduinos para resolver o problema em questão, bem como o desenvolvimento de um programa que possa exibir os dados coletados de forma gráfica e comparando estes dados com o banco de dados do INMET visando assim facilitar a compreensão destas informações para o usuário final.

## 5 Exemplos de dados coletados

A tabela 6 demonstra os dados coletados pela a mini-estação meteorológica.

*Tabela 6 Exemplo de Dados Coletados.*

DATA	HORA	TEMP. IN.	P.A.	ALTI.	TEMP. EX.	PLUVI.
"29/11/2017 ; 19:36:55 ; 26.60 ; 846.00 ; 91726.00 ; 25.50 ; 0.00 ;"						
"29/11/2017 ; 19:36:56 ; 26.60 ; 845.63 ; 91732.00 ; 25.56 ; 0.00 ;"						
"29/11/2017 ; 23:51:17 ; 25.50 ; 824.99 ; 91963.00 ; 24.31 ; 0.25 ;"						
"29/11/2017 ; 23:51:18 ; 25.50 ; 824.90 ; 91965.00 ; 24.38 ; 0.25 ;"						
"30/11/2017 ; 08:56:52 ; 23.40 ; 830.76 ; 91901.00 ; 21.56 ; 0.25 ;"						
"30/11/2017 ; 08:56:53 ; 23.50 ; 830.58 ; 91893.00 ; 21.56 ; 0.25 ;"						
"30/11/2017 ; 08:56:54 ; 23.50 ; 830.58 ; 91896.00 ; 21.56 ; 0.50 ;"						
"30/11/2017 ; 08:56:55 ; 23.50 ; 831.03 ; 91896.00 ; 21.56 ; 0.50 ;"						
"30/11/2017 ; 08:58:42 ; 23.40 ; 831.93 ; 91889.00 ; 21.50 ; 0.75 ;"						
"30/11/2017 ; 03:20:31 ; 24.70 ; 842.39 ; 91769.00 ; 23.06 ; 0.00 ;"						
"30/11/2017 ; 03:20:32 ; 24.70 ; 842.39 ; 91771.00 ; 23.06 ; 0.00 ;"						
"01/12/2017 ; 13:27:51 ; 23.80 ; 841.48 ; 91774.00 ; 23.94 ; 0.25 ;"						
"01/12/2017 ; 13:27:52 ; 23.80 ; 841.84 ; 91772.00 ; 23.94 ; 0.25 ;"						
"02/12/2017 ; 06:10:38 ; 22.90 ; 850.96 ; 91669.00 ; 21.75 ; 4.25 ;"						
"02/12/2017 ; 06:10:39 ; 22.90 ; 850.87 ; 91676.00 ; 21.75 ; 4.25 ;"						
"02/12/2017 ; 06:10:40 ; 22.90 ; 851.77 ; 91668.00 ; 21.75 ; 4.50 ;"						
"02/12/2017 ; 06:10:41 ; 22.90 ; 850.96 ; 91669.00 ; 21.75 ; 4.50 ;"						

*Fonte: Autoria própria.*

A fim de coletar o maior número de dados possível a mini-estação persiste num cartão SDCard os dados a cada segundo.

Para que o exemplo não ficasse extenso foram removidos os dados repetidos e exibidos apenas as variações de medida para mostrar o seu funcionamento.

## 6 Conclusão

Mediante as pesquisas bibliográficas e uso de tecnologias como linguagem de programação C, plataforma de prototipagem Arduino e sensores foi criado o protótipo de mini-estação meteorológica.

O desenvolvimento do projeto se mostrou complexo devido às alterações da forma de exibir os dados de cada sensor de forma individual, para uma forma de exibição conjunta, para assim poder reunir os dados em um só objeto e deste modo persistir os dados obtidos em um cartão de memória SDCard.

Sua IDE tornou a programação mais fácil por ser interligada diretamente ao *bootloader* do processador do Arduino, dispensando assim acesso aos binários do Arduino a cada Upload de código desenvolvido.

Mesmo usando a linguagem de programação C, devido sua limitação de memória, alguns recursos inerentes a linguagem em questão não estão presentes como, por exemplo, o *try catch*.

Aferindo os resultados da mini-estação montada, foi possível chegar a conclusão que existe uma exceção a ser tratada com relação ao sensor BMP180 e o pluviômetro de báscula.

Com o recurso *try catch* ativo, seria possível poder sanar o problema através do mesmo, porém na sua ausência se faz necessário uma análise detalhada na biblioteca do BMP180 para identificar qual ou quais métodos internos estão sendo responsáveis pelo conflito que reduz a precisão dos dados obtidos pelo pluviômetro.

Com exceção do conflito encontrado entre o sensor BMP180 e o pluviômetro de báscula, o projeto se mostrou eficaz na monitoração dos dados climáticos para o qual fora projetado.



## 7 Referências

ARDUINO.CC. **Arduino Duemilanove**. 2009. Disponível em: <<https://www.Arduino.cc/en/Main/ArduinoBoardDuemilanove>>. Acesso em: 01 out. 2017.

BASCONCELLO FILHO, Daniel. Curso de Arduino – Aula 3 – O Ambiente de Desenvolvimento Integrado. 2016. Disponível em: <<http://www.robotizando.com.br/pt-br/curso-de-Arduino-aula-3-ambiente-de-desenvolvimento-integrado/>>. Acesso em: 07 dez. 2017.

BASCONCELLO FILHO, Daniel. Curso de Arduino – Aula 3 – O Ambiente de Desenvolvimento Integrado. 2016b. Disponível em: <<http://www.robotizando.com.br/pt-br/curso-de-Arduino-aula-3-ambiente-de-desenvolvimento-integrado/2/>>. Acesso em: 07 dez. 2017.

BASCONCELLO FILHO, Daniel. Curso de Arduino – Aula 3 – O Ambiente de Desenvolvimento Integrado. 2016c. Disponível em: <<http://www.robotizando.com.br/pt-br/curso-de-Arduino-aula-3-ambiente-de-desenvolvimento-integrado/3/>>. Acesso em: 07 dez. 2017.

BOECKER-SYSTEMELEKTRONIK.DE. **Roboter programmieren mit Arduino (Teil 1: Konzept und Inbetriebnahme)**. 2017. Disponível em: <<https://www.boecker-systemelektronik.de/Seite/-/Kategorie-1/Roboter-programmieren-mit-Arduino-Teil-1-1>>. Acesso em: 01 out. 2017.

EXPOSTI, Karen Degli. **Climatologia**. 2013. Disponível em: <<https://www.infoescola.com/ciencias/climatologia/>>. Acesso em: 26 fev. 2018.

GOMBA, Davide. **Nice drawings of the Arduino UNO and Mega 2560**. 2011. Disponível em: <<http://blog.Arduino.cc/2011/01/05/nice-drawings-of-the-Arduino-uno-and-mega-2560/>>. Acesso em: 01 out. 2017.

GROENEVELD, Roland. **BeagleBone Black \$45 Linux Computer; Enclosure Coming Soon**. 2013. Disponível em: <<https://www.logicsupply.com/explore/io-hub/beaglebone/>>. Acesso em: 01 dez. 2017.

INMET. **CLIMA**. [1909]. Disponível em: <<http://www.inmet.gov.br/portal/index.php?r=home/page&page=clima>>. Acesso em: 26 fev. 2018.

LIMA, Thiago. **Intel Galileo - Placa Arduino**. 2014. Disponível em: <<https://www.embarcados.com.br/intel-galileo/>>. Acesso em: 07 jan. 2018.

MARTINS, Ramiro. **RTC DS3231**. 2017. Disponível em: <<https://cluberobotica.wordpress.com/2017/04/25/rtc-ds3231/>>. Acesso em: 07 nov. 2017.

MONK, Simon. Programação com Arduino: Começando com sketches. Porto Alegre:

Bookman, 2013. 147 p. (Teckne).

\_\_\_\_\_. Programação com Arduino II: Passos avançados com sketches. Porto Alegre: Bookman, 2015. 247 p. (Tekne).

MULTILÓGICA-SHOP, Equipe da. ArduinoGuia Iniciante. 2009. Disponível em: <[https://multilogica-shop.com/download\\_guia\\_Arduino](https://multilogica-shop.com/download_guia_Arduino)>. Acesso em: 07 dez. 2017.

NATANAEL. **Comparação das três plataformas de prototipagem mais comuns.** 2013. Disponível em: <<http://www.blogdonatanael.com/2013/05/comparacao-das-tres-plataformas-de.html>>. Acesso em: 02 jan. 2018.

NAYLAMPMECHATRONICS. **Tutorial Arduino y memoria SD y micro SD.** 2016. Disponível em: <[http://www.naylampmechatronics.com/blog/38\\_Tutorial-Arduino-y-memoria-SD-y-micro-SD-.html](http://www.naylampmechatronics.com/blog/38_Tutorial-Arduino-y-memoria-SD-y-micro-SD-.html)>. Acesso em: 10 out. 2017.

PORTAL G1. Chuvas no RJ. **Chuva na Região Serrana é maior tragédia climática da história do país.** 2011. Disponível em: <<http://g1.globo.com/rio-de-janeiro/chuvas-no-rj/noticia/2011/01/chuva-na-regiao-serrana-e-maior-tragedia-climatica-da-historia-do-pais.html>>. Acesso em: 06 fev. 2018

SEMPREUPDATE. Conhecendo o Fritzing. 2017. Disponível em: <<https://sempreupdate.com.br/conhecendo-o-Fritzing/>>. Acesso em: 07 dez. 2017.

SOUZA, Fábio de. Arduino UNO. 2013. Disponível em: <<https://www.embarcados.com.br/Arduino-uno/>>. Acesso em: 29 dez. 2017.

STORE.ARDUINO.CC. **Arduino Nano.** 2017a. Disponível em: <<https://store.Arduino.cc/usa/Arduino-nano>>. Acesso em: 10 out. 2017.

STORE.ARDUINO.CC. **Arduino Mega 2560 Rev3.** 2017b. Disponível em: <<https://www.Arduino.cc/en/Main/ArduinoBoardMega>>. Acesso em: 07 out. 2017.

THOMSEN, Adilson. **O que é Arduino?** 2014. Disponível em: <<https://www.filipeflop.com/blog/o-que-e-Arduino/>>. Acesso em: 01 out. 2017.

UPTON, Eben. **Raspberry Pi 2 on sale now at \$35.** 2015. Disponível em: <<https://www.raspberrypi.org/blog/raspberry-pi-2-on-sale/>>. Acesso em: 07 nov. 2017.

## 8 Anexos

### 8.1 Código BMP 180

```
void setup_bmp(){
  if (!bmp.begin())
  {
    Serial.println("BMP180 sensor not found");
    while (1){
    }

  }
}

// fim do setup_bmp
//função para leitura da temperatura

int outro_bmp180(char tipo){
  if (tipo == 't' ){
    return bmp.readTemperature();
  }
  else if (tipo == 'a'){
    return bmp.readAltitude(101500);
  }
  else if (tipo == 'p') {
    return bmp.readPressure();
  }
}

String ler_bmp180(){
  char tmpBmp[6];
  char tmpBmpa[6];
  char tmpBmpp[6];
  char tempBmp[30];
  char *valorBmpt=dtostrf((bmp.readTemperature()),4, 2, tmpBmp);
  strcpy(tempBmp, valorBmpt);
  strcat(tempBmp, " ; ");
  char *valorBmpa=dtostrf((bmp.readAltitude(101500)),4, 2, tmpBmpa);
  strcat(tempBmp,valorBmpa);
  strcat(tempBmp, " ; ");
  char *valorBmpp=dtostrf((bmp.readPressure()),4, 2, tmpBmpp);
  strcat(tempBmp,valorBmpp);
  strcat(tempBmp, " ; ");

  return tempBmp;
}
```

```
}
```

## 8.2 Código DS18B20

```
void setup_ds(void) {  
  
}
```

```
float temp(void) {  
byte i;  
byte present = 0;  
byte type_s;  
byte data[12];  
byte addr[8];  
float celsius, fahrenheit;  
celsius = 0;  
if ( !ds.search(addr)) {  
ds.reset_search();  
return temp();  
}
```

```
ds.reset();  
ds.select(addr);  
ds.write(0x44);
```

```
present = ds.reset();  
ds.select(addr);  
ds.write(0xBE);
```

```
for ( i = 0; i < 9; i++) {  
data[i] = ds.read();  
}  
int16_t raw = (data[1] << 8) | data[0];  
if (type_s) {  
raw = raw << 3; // 9 bit resolution default  
if (data[7] == 0x10) {  
raw = (raw & 0xFFF0) + 12 - data[6];  
}  
}  
else {  
byte cfg = (data[4] & 0x60);
```

```
if (cfg == 0x00) raw = raw & ~7; // 9 bit resolution, 93.75 ms  
else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms  
else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms  
///// default is 12 bit resolution, 750 ms conversion time  
}
```

```

celsius = (float)raw / 16.0;
return celsius;
}
String ler_ds(){
char data[6];
char tempe[16];
char *valor=dtostrf(temp(),4, 2, data);
strcpy(tempe, valor);
strcat(tempe," ; ");
return tempe;
}

```

### 8.3 Código do Pluviômetro deBáscula

```

void setup_pl(){
strcpy(valorPl, "0.00 ;" );
}

String ler_pl(){
val = digitalRead(REED);

if ((val == LOW) && (old_val == HIGH)){
delay(10);

REEDCOUNT = REEDCOUNT + 1;

old_val = val;

char tmp_pl[6];

char *valorP=dtostrf((REEDCOUNT*0.25),4, 2, tmp_pl);

strcpy(valorPl, valorP);
strcat(valorPl," ;");
}

else {

old_val = val;

```

```

}

return valorPl;
}

```

## 8.4 Código do REAL TIME CLOCK DS3231

```

byte decToBcd(byte valRtc){
return( (valRtc/10*16) + (valRtc%10) );
}
byte bcdToDec(byte valRtc){
return( (valRtc/16*10) + (valRtc%16) );
}
void setup_rtc(){
// set o valor inicial para o rtc:
// DS3231 seconds, minutes, hours, day, date, month, year
//setDS3231time(00,11,00,3,14,9,17);
}

void setDS3231time(byte second, byte minute, byte hour, byte
dayOfWeek, byte
dayOfMonth, byte month, byte year){
Wire.beginTransmission(DS3231_I2C_ADDRESS);
Wire.write(00);
Wire.write(decToBcd(second));
Wire.write(decToBcd(minute));
Wire.write(decToBcd(hour));
Wire.write(decToBcd(dayOfWeek));
Wire.write(decToBcd(dayOfMonth));
Wire.write(decToBcd(month));
Wire.write(decToBcd(year));
Wire.endTransmission();
}

void readDS3231time(byte *second,
byte *minute,
byte *hour,
byte *dayOfWeek,
byte *dayOfMonth,
byte *month,
byte *year){
Wire.beginTransmission(DS3231_I2C_ADDRESS);
Wire.write(0);
Wire.endTransmission();
Wire.requestFrom(DS3231_I2C_ADDRESS, 7);
*second = bcdToDec(Wire.read() & 0x7f);
*minute = bcdToDec(Wire.read());
*hour = bcdToDec(Wire.read() & 0x3f);
*dayOfWeek = bcdToDec(Wire.read());
*dayOfMonth = bcdToDec(Wire.read());
*month = bcdToDec(Wire.read());
*year = bcdToDec(Wire.read());
}

```

```

void displayTime(){
byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
readDS3231time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth,
&month,
&year);

```

```

Serial.print(hour, DEC);
displayed
Serial.print(":");
if (minute<10){
Serial.print("00");
}
Serial.print(minute, DEC);
Serial.print(":");
if (second<10){
Serial.print("00");
}
Serial.print(second, DEC);
Serial.print(" ");
Serial.print(dayOfMonth, DEC);
Serial.print("/");
Serial.print(month, DEC);
Serial.print("/");
Serial.print(year, DEC);
Serial.print(" Dia da Semana: ");
switch(dayOfWeek){
case 1:
Serial.println("Domingo");
break;
case 2:
Serial.println("Segunda");
break;
case 3:
Serial.println("Terça");
break;
case 4:
Serial.println("Quarta");
break;
case 5:
Serial.println("Quinta");
break;
case 6:
Serial.println("Sexta");
break;
case 7:
Serial.println("Domingo");
break;
}
}

```

```

String ler_rtc(){
byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;

```



```
readDS3231time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth,
&month,
&year);
```

```
char temp[5];
char TimeDate[25];
if(dayOfMonth<=9){
strcpy(TimeDate, itoa( 0, temp, 10));
};
strcat(TimeDate, itoa(dayOfMonth, temp, 10));
strcat(TimeDate, "/");
```

```
if(month<=9){
strcat(TimeDate, itoa( 0, temp, 10));
};
strcat(TimeDate, itoa( month, temp, 10));
strcat(TimeDate, "/");
```

```
strcat(TimeDate, itoa( 20, temp, 10));
strcat(TimeDate, itoa( year, temp, 10));
strcat(TimeDate, " ; ");
```

```
if(hour<=9){
strcat(TimeDate, itoa( 0, temp, 10));
};
strcat(TimeDate, itoa( hour, temp, 10));
strcat(TimeDate, ":");
if(minute<=9){
strcat(TimeDate, itoa( 0, temp, 10));
};
strcat(TimeDate, itoa( minute, temp, 10));
strcat(TimeDate, ":");
if(second<=9){
strcat(TimeDate, itoa( 0, temp, 10));
};
strcat(TimeDate, itoa( second, temp, 10));
strcat(TimeDate, " ; ");
return TimeDate;
```

```
}
```

## 8.5 Código Para SDCard:

```
void setup_sd(void)
{
inicia_SD();
}
```

```
//Grava dados no cartao SD
void grava_SD()
```

```

{
  abre_arquivo_gravacao("estação.txt");

  fecha_arquivo();
}

//Correcao para imprimir "00" ao inves de "0" caso
//o valor seja menor do que 10
void printI00(int val, char delim)
{
  if (val < 10) Serial << '0';
  Serial << _DEC(val);
  if (delim > 0) Serial << delim;
  return;
}

void inicia_SD()
{
  pinMode(CS_PIN, OUTPUT);

  if (SD.begin())
  {
  }
  else
  {
    return;
  }
}

int abre_arquivo_gravacao(char filename[])
{
  file = SD.open(filename, FILE_WRITE);

  if (file)
  {
    return 1;
  }
  else
  {
    return 0;
  }
}

void fecha_arquivo()
{
  if (file)
  {
    file.close();
  }
}

```

## 8.6 Classe MAIN Para Chamar Todos os Sensores Ligados

```
#include <Adafruit_BMP085.h>
#include <OneWire.h>
OneWire ds(7);

// includes do cartao sd

#include <DS3232RTC.h>
#include <Streaming.h>
#include <Time.h>
#include <Wire.h>
#include <SD.h>
#include <SPI.h>

// ----- variaveis
Adafruit_BMP085 bmp;
#define BMP085_I2C_ADDRESS 0x77

// ----- cartao de memoria

int CS_PIN = 4;
File file;
String dataAtual;
String dataAnterior;
String informacoes;
int contReinicio=0;
int valorReinicio=300;
String chuvaPl;
String chuvaPlAnterior;
int REEDCOUNT = 0;
int val = 0;
int old_val = 0;
char valorPl[15];

//-----pluviometro
const int REED = 6;

// ----- rtc
#define DS3231_I2C_ADDRESS 0x68
void setup() {

  Serial.begin(9600);
  setup_bmp();
  setup_sd();
  setup_rtc();
  setup_pl();
  dataAtual=ler_rtc();
  dataAnterior=ler_rtc();
  chuvaPlAnterior = ler_pl();
}

void loop() {
```

```

dataAtual=ler_rtc();
chuvaPl= ler_pl();

informacoes=dataAtual + ler_bmp180() + ler_ds() + chuvaPl;
if(dataAtual!=dataAnterior){
if(chuvaPl!=chuvaPlAnterior){

contReinicio=0;

}
else{

contReinicio ++;

}
chuvaPlAnterior = chuvaPl;
if( contReinicio >= valorReinicio){

REEDCOUNT = 0;
contReinicio = 0;
val = 0;

old_val = 0;
strcpy(valorPl, "0.00 ;" );

}

Serial.println(informacoes );
int flag=abre_arquivo_gravacao("estacao.txt");

if(flag==1){
Serial.println("Gravando no Sd");
file.println(informacoes);
fecha_arquivo();

}
else{
Serial.println("Nao Gravando no Sd");
}
}
dataAnterior=dataAtual;
}
}

```