Temperature Analyzes Interfaces e Periféricos

André Thiago Borghi Couto, *Graduando, UFES*
Bruno Calmon Barreto... ..., *Graduando, UFES*
Carlos Humberto Brito, *Graduando, UFES*

Resumo—The main applications for the interfaces are sensors, and for prototyping we have the Arduino, simple and easy to assemble, to get parts, so the work can be done in several ways, using the interfaces between computer and device, or device and peripherals, the peripherals are the most important part, representing the exchange of information from the environment with the discretization for reading and computational analysis, thus creating a hierarchy that brings us the possibility of seeing what is in our reality.

Index Terms—Computer Science, Arduino, Comunnications, Interfaces and Peripherals, LaTeX, Ciência da Computação, Comunicações, Interfaces e Periféricos.

1 Introdução

O Ambiente tem muitas informações a serem estudadas, que podem ser utilizadas para ajudar a controlar certas ações de nosso cotidiano, então este trabalho tem o foco de promover uma análise da temperatura e umidade ambiente, sendo criado um log com as informações obtidas, também sendo realizada a apresentação da temperatura em tempo real. Ao decorrer deste relatório teremos a descrição dos componentes, dos códigos e das integrações necessárias.

2 Descrição dos objetivos funcionais do trabalho

Neste trabalho temos o objetivo de demonstrar a montagem e codificação para um sensor de temperatura em conjunto com um Arduíno, utilizando meios para armazenamento interno (Gerar estatísticas) e exibição sem conexão física com o aparelho para visualizar em tempo real as informações geradas. Opcionalmente foi adicionado um módulo de audível para avisos sonoros de elevação ou decaimento da temperatura.

3 Especificação da infraestrutura

3.1 O Arduíno

O hardware utilizado foi uma placa de Arduíno UNO, no qual foram utilizadas as seguintes portas, sendo elas definidas pelas bibliotecas que o Arduíno ou o dispositivo entrega para o uso das funções que o hardware está preparado para executar.

- *A.T.B. Couto é graduando em ciência da computação, na UFES, campus CEUNES, em São Mateus - ES.*
  E-mail: andreww.max@hotmail.com
- *C.H. Brito é graduando em ciência da computação, na UFES, campus CEUNES, em São Mateus - ES.*
  E-mail: carloshumbertojr@hotmail.com
- *B.C. Barreto é graduando em ciência da computação, na UFES, campus CEUNES, em São Mateus - ES.*
  E-mail: brunobc@gmail.com

Trabalho escrito em 1 de julho de 2019;

- De comunicação:
  1) Pinos do sensor de Temperatura e Umidade:
     - Data = 2
  2) Pinos Bluetooth para comunicação serial:
     - RX = 5
     - TX = 4
  3) Pino Buzzer para os sons:
     - Buz = 3
  4) Pino CS do Cartão microSD para armazenamento:
     - SS_PIN = 10;
     - MOSI_PIN = 11;
     - MISO_PIN = 12;
     - SCK_PIN = 13;
- De alimentação:
  1) 2x 5 volts;
  2) 1x ground;

3.2 Temperatura e Umidade

O sensor de temperatura e umidade DHT11, é um sensor simples para manipulá-lo, sendo a biblioteca **SDHT** utilizada para realizar as leituras para o projeto. Ela consiste na instanciação de um objeto SDHT que possui as funções de leitura, em graus Celsius (ºC) ou Fahrenheit (ºF). A pinagem utilizada é definida por 1, juntamente com os de alimentação 3.1. Uma peculiaridade identificada no projeto é que ao utilizarmos uma protoboard, temos mais oscilação da voltagem, pois o sensor basicamente faz uma leitura do decaimento da voltagem ambiente e envia digitalmente para a porta *Data*, precisamos sempre de controlar a voltagem de entrada do sensor, para o mais estável possível, sendo assim recomendamos utilizar uma porta direta na placa do

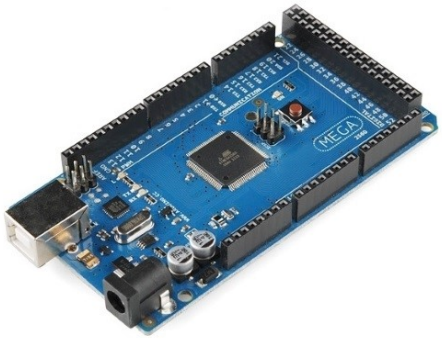

Figura 1. Arduíno Mega

Arduíno, ao invés de fazer a ponte na protoboard. Também salientamos que entrada do Arduíno sofre dos mesmos problemas, então com o intuito de evitar leituras muito dispersas, recomendamos sempre utilizar uma porta USB de um computador ou notebook.

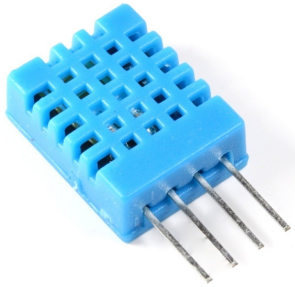

Figura 2. Sensor de Temperatura

3.3 Buzzer

O buzzer é uma simples peça de emissão sonora, também faz parte do kit básico do Arduíno, que utiliza apenas uma porta digital 3 e uma porta ground. A função tone faz a execução de funções para emissão de sons com diferentes entonações com tempo delimitado.

Figura 3. Buzzer

3.4 Cartão microSD

O Cartão MicroSD, é simples de manipular, pertencendo a biblioteca **SD**, tem o objetivo de armazenar dados em um arquivo simples de texto, porém o cartão deve ser formatado preferencialmente em FAT16, mas FAT32 também é aceito, uma peculiaridade da biblioteca é que funciona em modo **8.3** que é um padrão antigo definido para aceitar arquivos com o tamanho do nome específico de *[XXXXXXXX.XXX]*, assim sendo necessário que ao escolher o nome do arquivo passado para a função de abertura, ele esteja neste formato. Para fazer a inserção de dados no cartão é necessário abrir o arquivo com a função *open*, em modo de leitura, e a função *print* do objeto instanciado da biblioteca, e por fim fechar o arquivo com *close*

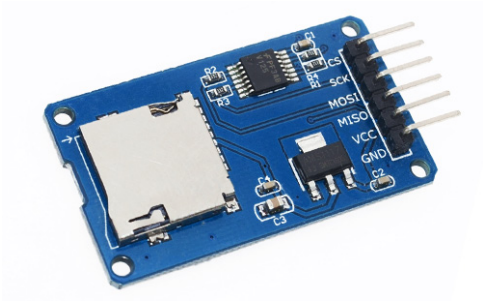

Figura 4. Módulo de microSD

3.5 Bluetooth

Neste projeto o Bluetooth é utilizado de forma serial, enviando dados para uma conexão simples com outro aparelho, pertencendo a biblioteca **SoftwareSerial**, que faz os tratamentos necessários para o envio das informações no formato serial correto, além de permitir a conexão com outros dispositivos. Pelo fato do Arduíno ser simples, temos a mesma semelhança no envio das informações com a *print* da instancia do objeto da sua Classe, porém contendo funções para descobrir se existe conexão, com o *avaliable*;

Figura 5. Módulo de Bluetooth

4 ESPECIFICAÇÃO DO SOFTWARE

Para este projeto utilizamos várias bibliotecas contidas no banco de dados do Arduíno, que são publicadas na comunidade aberta, e disponíveis para download no Gerenciador de Bibliotecas. Então conseguimos realizar os feitos presentes no trabalho, sendo de baixa complexidade, e aplicação para novas ideias.

4.1 Coleta de dados

A coleta de dados tem apenas que realizar a leitura do sensor de temperatura3.2, e adicioná-la as variáveis do código para as decisões serem tomadas, como no seguinte trecho de código.

```
// Inclui a biblioteca e declara o objeto da classe
#include "SDHT.h"
SDHT dht;
...
// Ativa a sincronização para o objeto receber as
  ↪  informações
    dht.broadcast(DHT11, 2)
...
// Retorna a temperatura e umidade no instante
    dht.celsius;
    dht.humidity;
...
```

Listing 1: Código Coleta de Dados

Através deste código temos a inclusão da biblioteca da classe e a declaração do objeto, em seguida a aquisição da temperatura e umidade relativa no instante que foi executada, pelo fato do Arduíno ser de uso continuo temos um loop, que fará o processo repetir enquanto estiver ligado.

4.2 Gerenciamento de arquivos

O gerenciamento de arquivos vem para organizar os dados que são gerados pelo sensor 3.2, e durante a execução temos a gravação em disco das informações no instante.

Nesta parte temos a inclusão da biblioteca, e em seguida os testes de abertura e instanciação do objeto, neste caso com o fluxo correto é gravado uma *String* parada pelo *print*, em seguida o arquivo é fechado.

4.3 Gerenciamento de Serialização

Neste contexto temos a serialização das informações para o dispositivo bluetooth, assim após a conexão ser estabelecida, temos o envio das informações para o outro aparelho. Após a inclusão da biblioteca, definição dos pinos e instanciação do objeto, então temos a sincronização do tempo de envio para aparelhos a serem conectados, então utilizamos o *print*, que é do mesmo principio do cartão, porém é continuo como um canal de envio.

```
// Inclusão da biblioteca, definindo o pino 10
  ↪  como de stream de dados e instanciação do
  ↪  objeto
#include <SD.h>
const int pinoSS = 10;
File myFile;
...
// Confirma se o cartão está disponível
    if (SD.begin()) {
// Abre o arquivo especificado em modo de escrita,
// caso não exista é criado.
        myFile = SD.open("TEMP.TXT", FILE_WRITE);
// Se aberto grava no cartão a String,
// Senão exibe o erro
        if (myFile){
            myFile.println(String("Temperatura e
              ↪  Umidade"));
        } else
            Serial.println("Erro ao Abrir Arquivo
              ↪  .txt");
// Fecha o arquivo
        myFile.close();
    }
```

Listing 2: Código Gerenciamento de Arquivos

```
// Inclui a biblioteca para comunicação Serial
  ↪  bluetooth
#include <SoftwareSerial.h>
// Define os Pinos Bluetooth
const int pinoRX = 5;
const int pinoTX = 4;
// Definição do objeto inicializado do bluetooth
SoftwareSerial hc06(pinoRX, pinoTX);
...
// Inicializa o sincronismo serial do bluetooth
    hc06.begin(9600);
// Envia para o bluetooth
    hc06.println(String("Temperatura e Umidade")
```

Listing 3: Código Gerenciamento de Serialização

5 DESCRIÇÃO DOS TESTES MONTADOS

Par testes, utilizamos 3 situações, cada modo tendo a visão de diferentes maneiras sobre a leitura da temperatura e da umidade, obtendo valores reais e que podem ser analisados de maneira objetiva em cada situação, e realizando o trabalho proposto com sucesso. Os testes foram realizados com o objetivo de retirar informações do ambiente, em 3 situações nas quais o Arduíno ficou ligado à tomada em um carregador, em uma situação de isolamento de outros dispositivos (offshore), simulando uma situação mais realista, onde o funcionamento é totalmente independente de outras plataformas, servindo apenas para consulta, como um servidor de dados local, no caso a conexão bluetooth era livre, mas é necessário uma senha do próprio módulo.

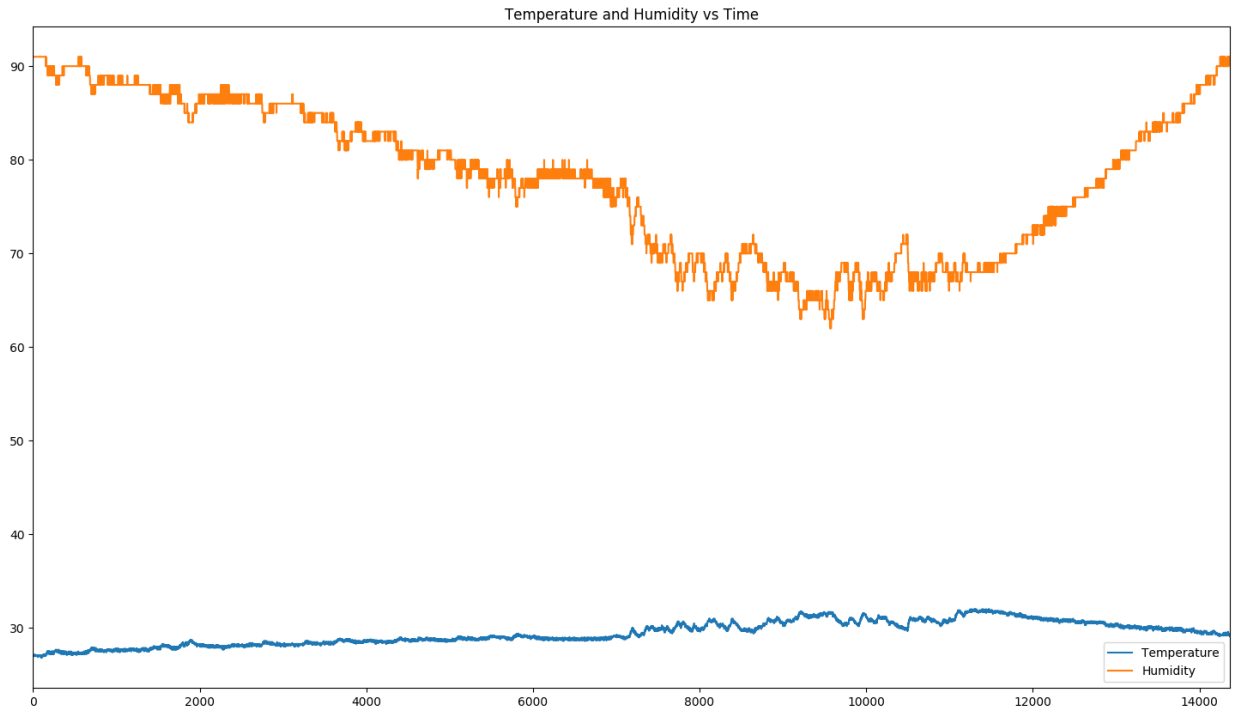

Figura 6. | **Temperatura e Umidade de 07h às 17h.**
O gráfico foi gerado a partir de dados de um local de ambiente aberto durante o dia, das 7h às 17h, vemos que a temperatura aumenta, enquanto a umidade diminui ao aproximar-se do meio dia, com uma maior concentração de calor por volta de 13:00h e 15:30h, e decaindo a partir do fim da tarde. Isso se deve pelo local ocorrer incidência de sol, contribuindo para a elevação da temperatura e alterando a umidade no local.

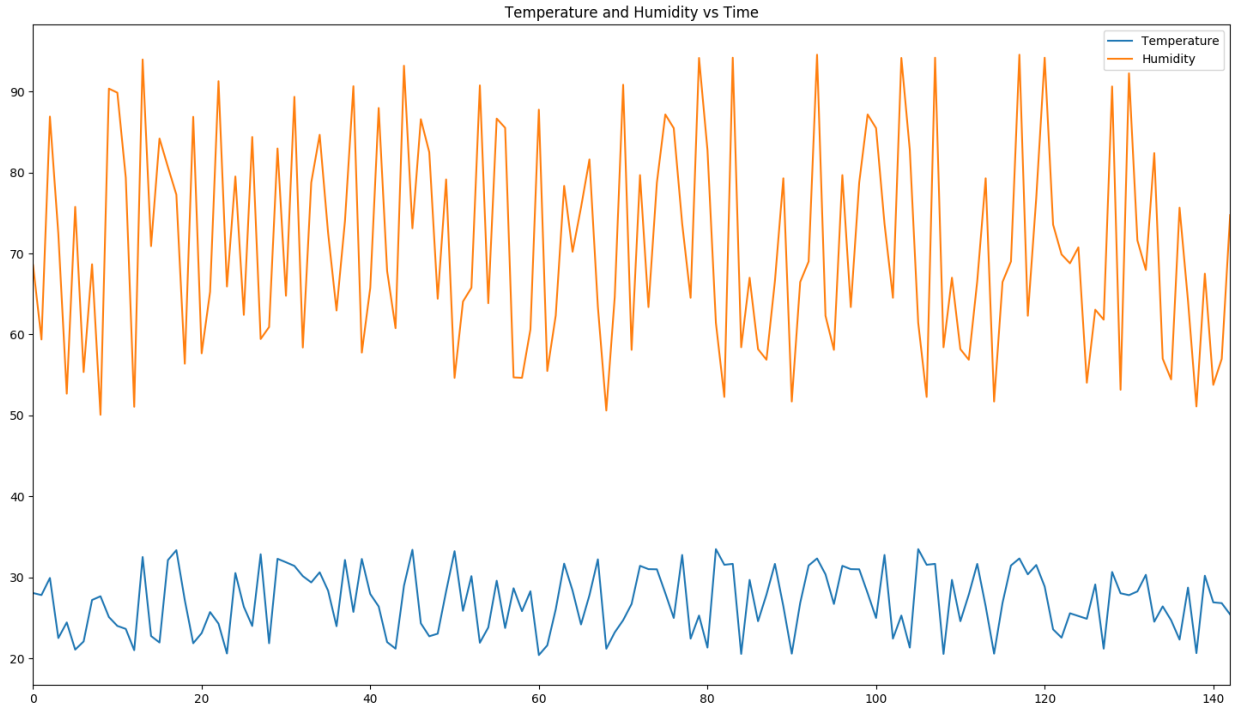

Figura 7. | **Análise do exaustor de um Notebook por 2 horas.**
Neste caso temos dados retirados de um local especifico, localizando o sensor próximo a saída do exaustor de um notebook, durante um uso severo, para obter as informações da temperatura de acordo com o uso. Foi feito um benchmarking, que faz testes com o a CPU e a GPU do notebook, gerando calor e por fim o sistema de ventilação acionado tenta reduzir a temperatura retirando o ar quente, sendo assim feita leitura da temperatura nessa situação. Vendo uma alta oscilação dos valores de temperatura e umidade, devido ao aumento e diminuição da exaustão do notebook.

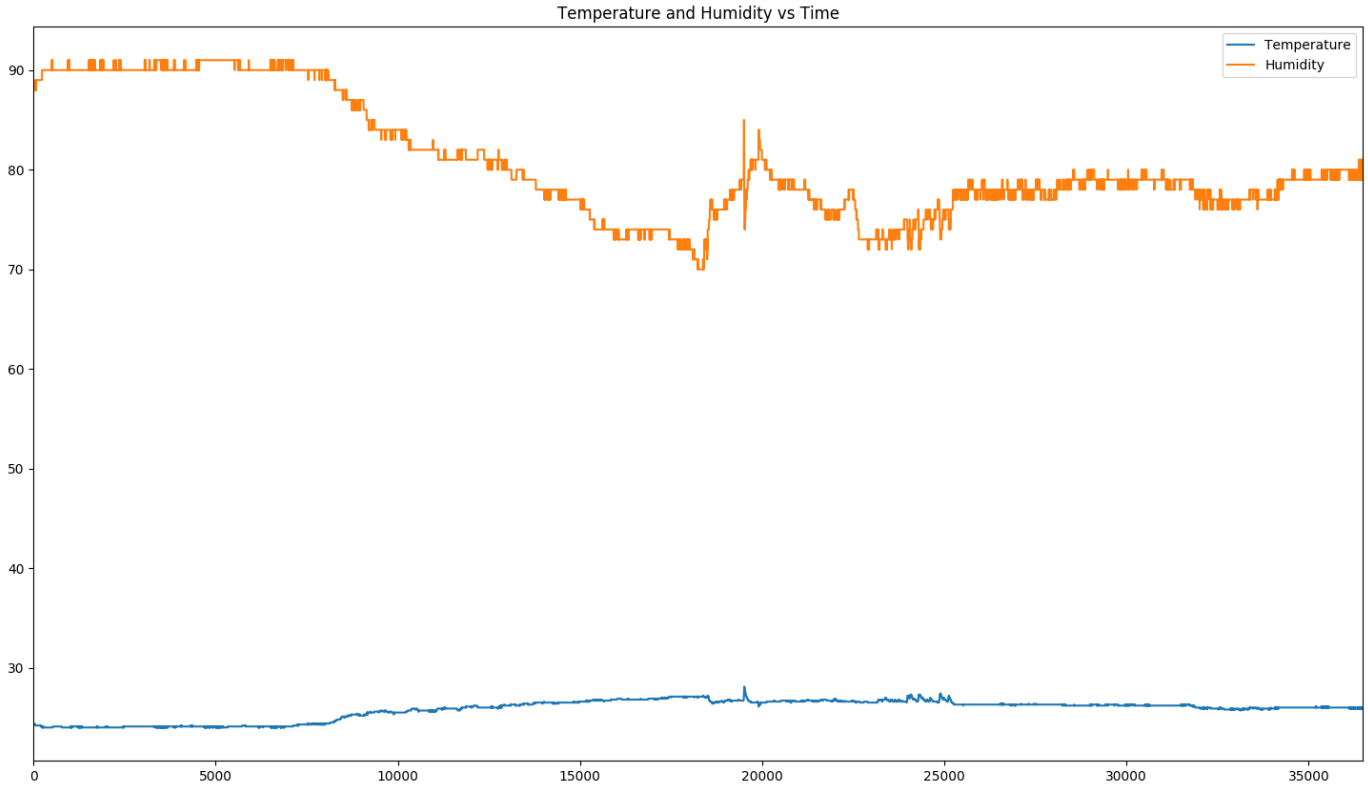

Figura 8. | **Análise longa dia 04/07/2019.**
Neste teste temos a leitura intermitente da temperatura do local, sendo capturada a cada 2 segundos durante aproximadamente 24 horas, em um ambiente coberto com circulação de ar. Obtemos um gráfico onde percebemos que não há muita diferença de temperatura, porém a humidade relativa, tem grandes oscilações principalmente nos momentos próximos às 12h.