INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE

CÂMPUS VENÂNCIO AIRES

MOSTRA VENÂNCIO-AIRENSE DE CULTURA E INOVAÇÃO (MOVACI)

BRENDA K. VARGAS DOS SANTOS

JÚNIOR HOMEM HENRIQUE

MATHEUS DAMASCENO DE LIMA

DATA LOGGER AUTOSSUSTENTÁVEL DE BAIXO CUSTO

PROF. TIAGO BAPTISTA NORONHA

ORIENTADOR

PROF. LEANDRO CÂMARA NORONHA

CO-ORIENTADOR

VENÂNCIO AIRES, 2016.

BRENDA K. VARGAS DOS SANTOS

JÚNIOR HOMEM HENRIQUE

MATHEUS DAMASCENO DE LIMA

DATA LOGGER AUTOSSUSTENTÁVEL DE BAIXO CUSTO

Projeto de pesquisa apresentado como requisito parcial para aprovação na Mostra Venâncio-Airense de Cultura e Inovação (MOVACI), evento promovido pelo Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense, Câmpus Venâncio Aires.

Professor Orientador: Tiago Baptista Noronha.

VENÂNCIO AIRES, 2016.

RESUMO

Pesquisadores de diversas áreas de conhecimento que necessitem coletar dados remotos para suas pesquisas, atualmente, podem contar com a utilização de *Data Loggers* comerciais. A aquisição destes equipamentos pode influenciar drasticamente nos custos de um projeto de pesquisa, uma vez que os dispositivos que encontramos no mercado atualmente possuem valores relativamente elevados. Parte desse alto custo deve-se ao fato de contarem com excelentes interfaces de entrada, proteção contra interferência eletromagnética, além de baterias com alta capacidade de carga e circuitos com baixo consumo de potência. Este projeto visa a pesquisa e a construção de um sistema de aquisição e armazenamento de dados utilizando tecnologias de código aberto, como também a utilização de fontes de alimentação sustentáveis, com o intuito de atender projetos de pesquisa que não necessitem de instrumentos tão precisos e exatos quanto os oferecidos comercialmente. Objetiva-se que a documentação produzida ao longo desta pesquisa reduza os custos e/ou viabilize projeto de pesquisas que necessitem utilizar o sensoriamento remoto como ferramenta de coleta de dados.

Palavras–chave: Sensoriamento remoto. Processamento digital de sinal. Instrumentação. Filtragem digital.

LISTA DE FIGURAS

Figura 1 - *Data Logger* Testo .....**Erro! Indicador não definido.**
Figura 2 - Arduino Mega 2560 R3 5
Figura 3 - Módulo *SD Card* 5
Figura 4 - Módulo *Ethernet Shield* W5100 5
Figura 5 - Sensor LM35.....6
Figura 6 - Sensor DHT11 6
Figura 7 - Sensor LDR 6
Figura 8 - Projeto PCB Sensores 7
Figura 9 – Página WEB para acesso remoto dos dados..... 7
Figuras 10, 11, 12 - Código Arduino 8
Figuras 13, 14, 15 - Código Arduino 9
Figuras 16, 17, 18 - Código Arduino 10
Figuras 19, 20, 21 - Código Arduino 11
Figuras 22, 23, 24 - Código Arduino 12
Figura 25 – Tabela de preços..... 13

SUMÁRIO

1 INTRODUÇÃO 1

2 OBJETIVOS 2

2.1 Objetivo geral 2

2.2 Objetivos específicos 2

3 REFERENCIAL TEÓRICO 3

4 METODOLOGIA 4

5 CONSIDERAÇÕES FINAIS 11

REFERÊNCIAS 12

1 INTRODUÇÃO

Diversos são os campos do conhecimento que necessitam coletar dados em seus projetos de pesquisa. Como exemplo, podemos citar as ciências agrícolas, geografia, biologia, entre outras [1][4]. Grande parte do orçamento de tais projetos é consumido por dispositivos de coleta e armazenamento de dados, pois as soluções encontradas no mercado apresentam elevado custo por conta da extrema exatidão e precisão de seus instrumentos. [2] Alguns dispositivos comerciais ainda contam com sistemas de proteção contra interferência eletromagnética, redutores de ruídos aditivos e outras técnicas que, por muitas vezes, fazem-se dispensáveis para projetos de determinadas naturezas.

Outro fator que contribui para o custo elevado na realização desses projetos, é a demanda por sistemas que funcionem utilizando baterias. Como muitas vezes, em experimentos, é necessário coletar dados por um grande período de tempo, *Data Loggers*[1] comerciais necessitam utilizar baterias de alta duração aliadas com sistema de baixo consumo de potência.

Este projeto visa reduzir o custo do dispositivo de coleta de dados através da utilização de técnicas de instrumentação mais simples e que, embora não sejam tão eficazes quanto as técnicas encontradas em dispositivos comerciais, sejam suficientes para projetos de certa natureza (e.g. quando o sinal a ser monitorado varie lentamente).

Também pretende-se explorar técnicas de alimentação renováveis, como é o caso da energia solar. Assim, mesmo para projetos que necessitem monitorar sinais por um grande período de tempo, não será necessário o uso de baterias de alta capacidade e/ou desenvolver uma eletrônica de baixo consumo de potência.

Pretende-se que o material desenvolvido nesta pesquisa permita que o dispositivo de aquisição e armazenamento de dados seja replicado, de maneira relativamente simples, para ser aplicado em projetos das mais diversas áreas do conhecimento. Viabilizando, assim, o estudo em determinadas áreas e contribuindo para o progresso da ciência nacional.

[1] Equipamento capaz de armazenar leituras de outros instrumentos de medição desde que estes transmitam a informação de alguma forma (analógica ou digital).

2 OBJETIVOS

2.1 Objetivo geral

Projetar e desenvolver uma plataforma de coleta de dados (*Data Logger*) alimentada por energia solar e de custo reduzido (quando comparada com soluções comerciais).

2.2 Objetivos específicos

- Estudar as técnicas de instrumentação;
- Estudar técnicas de alimentação renovável;
- Estudar os módulos de *hardware*[2] necessários para construção do sistema;
- Realizar o diagrama de blocos do sistema, identificando quais partes serão resolvidas em *hardware* e quais pontos serão resolvidos em *software*[3];
- Codificar os blocos do sistema capazes de coletar e armazenar as informações;
- Implementar filtros digitais capazes de mitigar os problemas acarretados devido a qualidade dos instrumentos utilizados;
- Desenvolver *software* capaz de ler as informações contidas na memória não volátil do equipamento;
- Desenvolver sistema mecânico capaz de comportar o circuito garantindo que o sistema possa ser exposto as intemperes;
- Desenvolver sistema sustentável de alimentação.

[2] Termo usado para designar as peças, circuitos, e componentes eletrônicos em geral.

[3] Sequência de instruções escritas para serem interpretadas por um computador com o objetivo de executar tarefas específicas.

3 REFERENCIAL TEÓRICO

Com o passar dos anos e o avanço na área de pesquisa para desenvolvimento de dispositivos tecnológicos, podemos notar o aumento de recursos que visam facilitar as nossas vidas e aumentar a efetividade de nossos trabalhos. Atualmente temos disponível no mercado vários dispositivos que tem como objetivo a leitura e o armazenamento de dados climáticos em locais mais específicos, como por exemplo em uma estufa ou em um projeto de pesquisa que necessite desses dados.

Segundo o site da produtora de equipamentos de medição Testo [6], o *Data Logger* Testo 176 P1 oferece ao usuário um sensor interno de pressão absoluta e duas conexões para sensores externos, um NTC[4] e um sensor de humidade capacitivo, suporte do parede, cadeado, pilhas e protocolo de calibração por um preço de R$ 3.569,00 o que é um custo demasiadamente alto para pequenos produtores ou grupos de pesquisa voltados para essa área.

Visamos com esse projeto construir um dispositivo que realize as mesmas funções que um *data logger* comercial, ou seja, realizar medições de dados climáticos armazenando os mesmos em um dispositivo de memória para que posteriormente sejam analisados e utilizados em prol do usuário que utilizá-lo, porém com o custo reduzido, assim possibilitando que um maior número de pessoas tenham acesso aos benefícios que este tipo de dispositivo proporciona. Nosso *Data Logger* possui diferenciais; um deles é a integração com a internet, possibilitando que o acesso seja feito pelo usuário remotamente; outro diferencial é o uso de fontes de alimentação renováveis, com o intuito de aumentar a autonomía da bateria e também diminuir a necessidade de manutenção, uma vez que a bateria totalmente carregada precisaria de dias para precisar ser carregada novamente.

Figura 1 - *Data Logger* Testo

[4] São resistores semicondutores sensíveis à temperatura e têm, de acordo com o seu tipo, um coeficiente resistência / temperatura negativo.

4 METODOLOGIA

Algumas condições foram utilizadas como base para o desenvolvimento deste projeto. Como principais, podemos citar: encontrar as grandezas que iríamos medir, levando em consideração quais seriam mais importantes para que, posteriormente, pudessem ser analisadas e utilizadas diante do propósito que o projeto aborda; realizar a escolha dos sensores que seriam utilizados e que deveriam atender os requisitos nos quais o projeto está submetido: medições confiáveis e que o custo final não fique elevado, a fim de que seja viável a implementação do mesmo.

Criamos um protótipo inicial para a realização de testes a fim de verificar o funcionamento. Até o momento, conseguimos desenvolver o armazenamento dos dados em *SD Card* e o envio dos mesmos para a Internet, o que já nos dá um bom controle do estado atual do projeto.

Utilizamos o microcontrolador[5] Arduino Mega 2560 R3 (Figura 1) este, por sua vez, é uma plataforma eletrônica de código aberto baseado em hardware e software de fácil utilização. É destinado para qualquer um fazer projetos interativos [7]. Além disso, possui um alto custo-benefício comparado aos demais microcontroladores encontrados atualmente no mercado. Também utilizamos um módulo *SD Card*[6] para Arduino (Figura 2), o módulo aceita cartões formatados em FAT16 ou FAT32[7], e utiliza a comunicação via interface SPI[8] por meio dos pinos MOSI, SCK, MISO e CS [8], possibilitando o armazenamento de uma grande quantidade de dados, sendo que será gravado arquivos de texto, que ocupam pouco espaço. Para a comunicação com a internet, utilizamos o Módulo *Ethernet Shield* W5100 (Figura 3). A coleta de dados foi feita a partir do uso de três sensores: LM35 (Figura 4) foi escolhido para ser o sensor de temperatura, em virtude de sua boa precisão e alta linearidade levando em conta o seu baixo custo de aquisição [9]. O DHT11 (Figura 5) é o sensor de umidade usado neste projeto, ele é o sensor que possui uma boa repetibilidade e também um curto tempo de resposta, obtendo leituras rápidas e valores relativamente precisos [10]. Como sensor de luminosidade foi utilizado o LDR (Figura 6), mesmo apresentando um tempo de resposta lento quando em comparação com outros sensores como fototransistores e fotodiodos, que têm a capacidade de perceber rápidas variações de luz (dezenas ou centenas de MegaHertz), ainda é um sensor viável para a aplicação dada neste projeto [11].

---

[5] Máquina capaz de buscar e executar instruções de programas alocados na sua memória.

[6] Dispositivo de memoria flash que pode armazenar informações digitais para o consumidor.

[7] São nomes de sistemas de arquivos (file sistems) utilizados por padrão em versoes antigas do sistema operacional Windows (Windows 98, por exemplo).

[8] Protocolo que permite a comunicação do microcontrolador com diversos outros componentes.

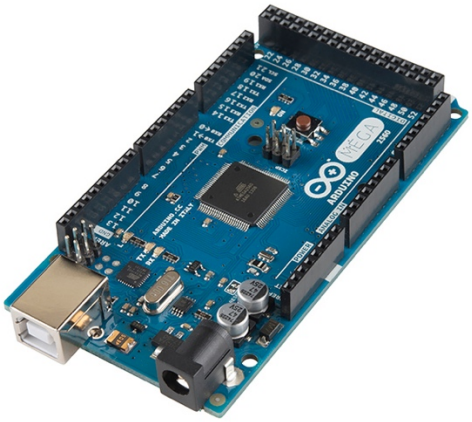

Figura 1 - Arduino Mega 2560 R3

Figura 2 - Módulo *SD Card*

Figura 3 - Módulo Ethernet Shield W5100

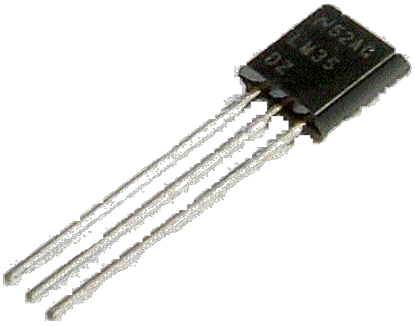

Figura 4 - Sensor LM35

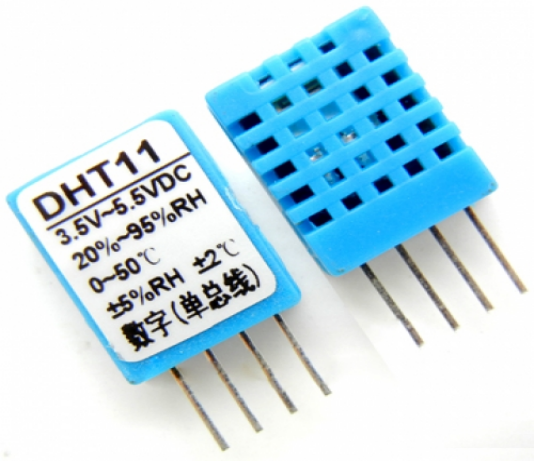

Figura 5 - Sensor DHT11

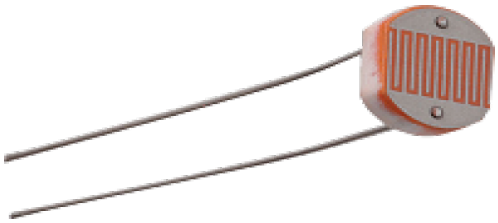

Figura 6 - Sensor LDR

Para a montagem desse protótipo, foi construído uma placa de circuito impresso para os sensores (Figura 8) utilizando o *software* de criação de *PCB's* [9] *Eagle* [12].

[9] Abreviação para "*printed circuit board*", ou seja, "placa de circuito impresso". É nela que os demais componentes dos dispositivos são soldados.

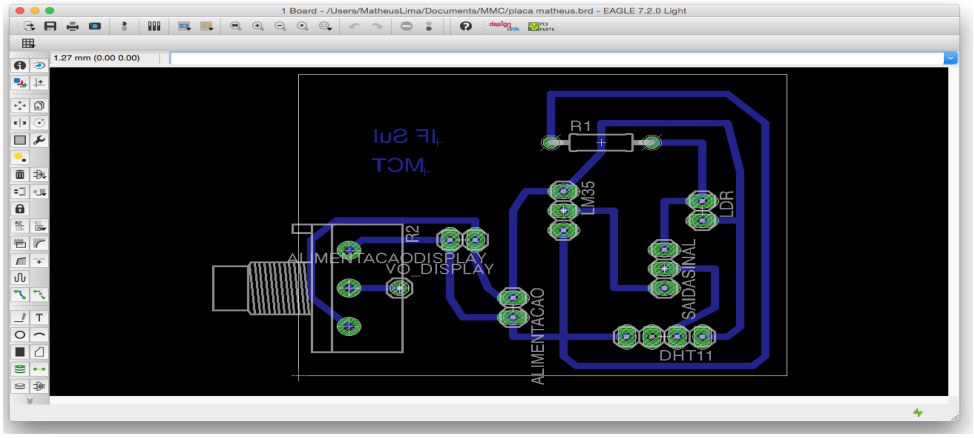

Figura 7 - Projeto PCB Sensores

Para o controle deste processo, foi desenvolvido um programa na IDE do Arduino (Figuras 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24). Para realizar as leituras dos sensores e apresentá-las no display LCD 16x2 e simultaneamente armazená-las no *SD Card*.

O desenvolvimento da página para acesso remoto (Figura 9) dos dados foi realizado em PHP[10] e HTML[11]. A formatação JSON[12][13] foi usada na comunicação entre o Data Logger e a página, onde os dados eram enviados do Data Logger para o IP fixo da rede, por meio do Módulo Ethernet SHIELD, e então a página faz a requisição dos dados de um determinado canal de leitura em um determinado dia, mês e ano.

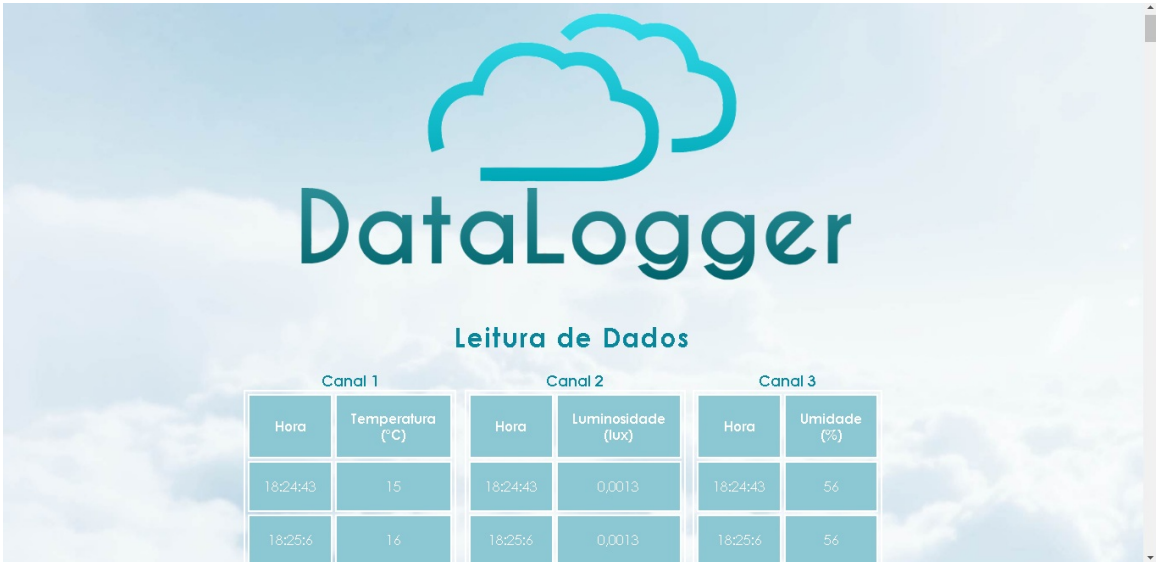

Figura 9 – Página WEB para acesso remoto dos dados

[10] Um acrônimo recursivo para **PHP**: *Hypertext Preprocessor*. É uma linguagem de script open source de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web.

[11] *Hypertext Markup Language* ou em português Linguagem de Marcação de Hipertexto. O HTML é a linguagem base da internet.

[12] *JavaScript Object Notation* - Notação de Objetos JavaScript. Uma formatação leve de troca de dados.

DataLogger | Arduino 1.6.8

DataLogger

```
#include "DHT.h" // biblioteca DHT11
#include "LiquidCrystal.h" // biblioteca para usar o display
#include "Limits.h"
#include "SPI.h"
#include "SD.h"
#include "Wire.h"
#include <Ethernet.h>

#define DHTPIN  A2
#define DHTTYPE DHT11
#define DS1307_ADDRESS 0x68

DHT dht(DHTPIN, DHTTYPE);

byte zero = 0x00;

// endereço mac do módulo
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};

const int sensorLuz  = A0; //Pino analógico em que o sensor de luz está conectado.
const int sensorTemp = A3; //Pino analógico em que o sensor de temperatura no sol está conectado.
//const int chipSelect = 7; //CS módulo SD

int valorSensorLuz  = 0;  //variável usada para ler o valor do sensor de luz.
int valorSensorTemp = 0;  //variável usada para ler o valor do sensor de temperatura.

long tempoAnteriorTemp = 0;
long tempoAnteriorLux = 0;
long tempoAnteriorUmid = 0;

int t = 1;
int l = 1;
int u = 1;

String hora = "";
String data = "";
String filename = "";

char c;
int cont = 0;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int segundos;
int minutos;
int horas;
int diadomes;
int diadasemana;
int mes;
int ano;

byte sol[8] = {
  B00100,
  B10001,
  B01110,
  B10001,
  B10001,
  B01110,
  B10001,
  B00100
};
byte nuvem[8] = {
  B00000,
  B00000,
  B01110,
  B11011,
  B11011,
  B01110,
  B00000,
  B00000
};

EthernetServer server(80);

void Mostrarelogio();

void setup() {
  pinMode(10, OUTPUT);
  digitalWrite(10, HIGH);

  //Inicializando o LCD e informando o tamanho de 16 colunas e 2 linhas
  Serial.begin(115200); // inicializa a serial
  while (!Serial) {
    ; // espera pela porta serial
  }
  dht.begin();
  Wire.begin();

  lcd.createChar(0, sol); // cria o desenho do sol no display
  lcd.createChar(1, nuvem); // cria o desenho da nuvem no display

  lcd.begin(16, 2); // inicializa o display
  lcd.setCursor(6, 0);
  lcd.print("Data");
  lcd.setCursor(0, 1);
  lcd.write(byte(1));
  lcd.setCursor(5, 1);
  lcd.print("Logger");
  lcd.setCursor(15, 1);
```

Figuras 10, 11 e 12 - Código Arduino

```
  lcd.write(byte(0));

  Serial.println("\t \t \t Data Logger Autossustentavel de Baixo Custo");
  Serial.print("Initializing SD card...");

  // verifica se o SD Card está presente
  if (!SD.begin(53)) {
    Serial.println("Falha ao encontrar SD Card");
  } else {
    Serial.println("Cartão encontrado");
  }

  if (Ethernet.begin(mac) == 0) {
    Serial.println("Falha em conectar à internet");
  }

  server.begin();
  Serial.print("Servidor está em: ");
  Serial.println(Ethernet.localIP());

  //SelecionaDataeHora();

  Mostrarelogio();
  delay(3500);
}


void loop() {
  String dataStringTemp = "";
  String dataStringLux = "";
  String dataStringUmid = "";

  // leitura sensor temperatura
  int mediaValorTemp  = 0;  //Variável usada para armazenar o menor valor da temperatura.
  int somaValorTemp = 0;
  int temp;

  somaValorTemp = 0;
  temp = 0;
  for (int k = 1; k <= 8; k++) {
    //Lendo o valor do sensor de temperatura.
    temp = analogRead(sensorTemp);

    somaValorTemp += temp;

    delay(250);
  }

  mediaValorTemp = somaValorTemp / 8;
  //Transformando valor lido no sensor de temperatura em graus celsius aproximados.
  mediaValorTemp = (500 * mediaValorTemp) / 1023;

  valorSensorTemp = mediaValorTemp;


  Serial.print("Temperatura: ");
  Serial.println(valorSensorTemp);

  dataStringTemp += String(valorSensorTemp);

  //Exibindo valor da leitura do sensor de temperatura no display LCD.
  lcd.clear();  //limpa o display do LCD.
  lcd.setCursor(2, 0);
  lcd.print("Temperatura: ");  //imprime a string no display do LCD.
  lcd.setCursor(6, 1);
  lcd.print(valorSensorTemp);
  lcd.write(B11011111); //Simbolo de graus celsius
  lcd.print("C");

  delay(1000);

  // Sensor de luz
  int mediaValorLuz  = 0;
  int somaValorLuz = 0;
  int luz = 0;

  for (int i = 1; i <= 8; i++) {
    //Lendo o valor do sensor de luz
    luz = analogRead(sensorLuz);

    somaValorLuz += luz;

    delay(250);
  }

  mediaValorLuz = somaValorLuz / 8;

  valorSensorLuz  = mediaValorLuz;

  double lux = pow((9.78 * valorSensorLuz) / (1E6 - 978.0 * valorSensorLuz), 5.0 / 3); // Calculo para conversão para LUX

  //Limpa o display
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("Luminosidade: ");  //imprime a string no display do LCD.
  lcd.setCursor(3, 1);
  lcd.print(lux, 4);
  lcd.setCursor(10, 1);
  lcd.print("Lux");

  Serial.print("Luminosidade: ");
```

Figuras 13, 14, 15 - Código Arduino

```
  Serial.print(lux, 4);
  Serial.println(" Lux");

  dataStringLux += String(lux, 4);

  delay(1000);

  // Sensor de humidade
  int mediaValorHumidade  = 0;
  int somaValorHumidade = 0;
  int humidade = 0;

  for (int j = 1; j <= 8; j++) {
    //Lendo o valor do sensor de humidade
    humidade = dht.readHumidity();

    somaValorHumidade += humidade;

    delay(250);
  }

  mediaValorHumidade = somaValorHumidade / 8;

  float h = mediaValorHumidade;

  lcd.clear();

  if (isnan(h))
  {
    Serial.println("Falha ao achar DHT11");
  }
  else
  {
    Serial.print("Umidade: ");
    Serial.print(h);
    Serial.println(" % ");

    dataStringUmid += String(h);

    lcd.setCursor(3, 0);
    lcd.print("Umidade do");
    lcd.setCursor(0, 1);
    lcd.print("Ambiente: ");
    lcd.print(h, 2);
    lcd.print(" % ");
    delay(1000);
  }

  SD.mkdir(data.c_str());

  if (millis() - tempoAnteriorTemp > 10000) {
    tempoAnteriorTemp = millis();

    filename = data + "/1.txt";

    File dataFile1 = SD.open(filename.c_str(), FILE_WRITE); // Abre o arquivo temp.txt no SD Card

    // verifica se o arquivo esta disponivel
    if (dataFile1) {
      dataFile1.print(dataStringTemp);
      dataFile1.print(",");
      dataFile1.print(hora);
      dataFile1.println("");
      dataFile1.close();

      Serial.print(dataStringTemp);
      Serial.println("");
    }
    // printa mensagem de erro
    else {
      Serial.println("Erro ao abrir temp.txt");
    }
  }

  if (millis() - tempoAnteriorLux > 10000) {
    tempoAnteriorLux = millis();

    filename = data + "/2.txt";

    File dataFile2 = SD.open(filename.c_str(), FILE_WRITE); // Abre o arquivo lux.txt no SD Card

    // verifica se o arquivo esta disponivel
    if (dataFile2) {
      dataFile2.print(dataStringLux);
      dataFile2.print(",");
      dataFile2.print(hora);
      dataFile2.println("");
      dataFile2.close();

      Serial.print(dataStringLux);
      Serial.println("");
    }
    // printa mensagem de erro
    else {
      Serial.println("Erro ao abrir lux.txt");
    }
  }

  if (millis() - tempoAnteriorUmid > 10000) {
```

Figuras 16, 17, 18 - Código Arduino

```
303     tempoAnteriorUmid = millis();
304
305     filename = data + String("/3.txt");
306     File dataFile3 = SD.open(filename.c_str(), FILE_WRITE); // Abre o arquivo umidade.txt no SD Card
307     Serial.println(filename);
308
309     // verifica se o arquivo esta disponivel
310     if (dataFile3) {
311       dataFile3.print(dataStringUmid);
312       dataFile3.print(",");
313       dataFile3.print(hora);
314       dataFile3.println("");
315       dataFile3.close();
316
317       Serial.print(dataStringUmid);
318       Serial.println("");
319     }
320     // printa mensagem de erro
321     else {
322       Serial.println("Erro ao abrir umid.txt");
323     }
324   }
325   Mostrarelogio();
326   Serial.println(hora);
327
328   String HTTP_req = "";
329
330   EthernetClient client = server.available();
331   if (client) {
332     Serial.println("Novo cliente");
333     // um http request acaba com um blank line
334     boolean currentLineIsBlank = true;
335     while (client.connected()) {
336       if (client.available()) {
337         c = client.read();
338
339         if (cont == 1) {
340           HTTP_req += c;
341         }
342
343         if (c == ' ') {
344           cont ++;
345         }
346
347         Serial.write(c);
348         // se chegou ao fim da linha (recebeu um caractere de nova linha)
349         // e a linha estiver em branco, o pedido http terminou,
350         // assim pode enviar uma resposta
351         if (c == '\n' && currentLineIsBlank) {
352           // enviar um cabeçalho de resposta HTTP padrão
353           client.println("HTTP/1.1 200 OK");
354           client.println("Content-Type: application/json");
355           client.println("Connection: close");  // a conecçao vai ser fechada depois da conclusão da resposta
356           client.println();
357           Serial.println(HTTP_req);
358           cont = 0;
359           HTTP_req.trim();
360           HTTP_req += ".txt";
361           File pag = SD.open(HTTP_req.substring(1).c_str(), FILE_READ);
362           Serial.println(HTTP_req.substring(1));
363           char pagi;
364           pagi = pag.read();
365           bool fimLinha = true;
366           client.print("{ \"dados\":[");
367           while (pagi != -1) {
368             if (fimLinha) {
369               client.print("{ \"dado\":");
370               fimLinha = false;
371             }
372             if (pagi == ',') {
373               client.print(",\"hora\":\"");
374             } else if (pagi == '\n') {
375               client.print("\"}");
376               fimLinha = true;
377             } else if (pagi != '\r') {
378               client.print(pagi);
379             }
380             pagi = pag.read();
381             if (fimLinha && pagi != -1) {
382               client.print(",");
383             }
384
385           }
386           client.print("]}");
387           break;
388         }
389       }
390       if (c == '\n') {
391         // you're starting a new line
392         currentLineIsBlank = true;
393       } else if (c != '\r') {
394         // you've gotten a character on the current line
395         currentLineIsBlank = false;
396       }
397     }
398   }
399   // give the web browser time to receive the data
400   delay(1);
401   // close the connection:
```

Figuras 19, 20, 21 - Código Arduino

```
  client.stop();
  Serial.println("client disconnected");
  Mostrarelogio();
}

byte ConverteParaBCD(byte val) { //Converte o número de decimal para BCD
  return ( (val / 10 * 16) + (val % 10) );
}

byte ConverteparaDecimal(byte val)  { //Converte de BCD para decimal
  return ( (val / 16 * 10) + (val % 16) );
}

void Mostrarelogio()
{
  Wire.beginTransmission(DS1307_ADDRESS);
  Wire.write(zero);
  Wire.endTransmission();
  Wire.requestFrom(DS1307_ADDRESS, 7);
  segundos = ConverteparaDecimal(Wire.read());
  minutos = ConverteparaDecimal(Wire.read());
  horas = ConverteparaDecimal(Wire.read() & 0b111111);
  diadasemana = ConverteparaDecimal(Wire.read());
  diadomes = ConverteparaDecimal(Wire.read());
  mes = ConverteparaDecimal(Wire.read());
  ano = ConverteparaDecimal(Wire.read());
  //Mostra a data no Serial Monitor

  data =  String();
  data += String(ano);
  data += String("/");
  data += String(mes);
  data += String("/");
  data += String(diadomes);

  if (horas < 10) hora += "0";
  hora = String(horas);
  hora += String(":");
  if (minutos < 10) hora += "0";
  hora += String(minutos);
  hora += String(":");
  if (segundos < 10) hora += "0";
  hora += String(segundos);
}

void SelecionaDataeHora()   //Seta a data e a hora do DS1307
{ byte segundos = 00; //Valores de 0 a 59
  byte minutos = 10; //Valores de 0 a 59
  byte horas = 20; //Valores de 0 a 23
  byte diadasemana = 2; //Valores de 0 a 6 - 0=Domingo, 1 = Segunda, etc.
  byte diadomes = 23; //Valores de 1 a 31
  byte mes = 8; //Valores de 1 a 12
  byte ano = 16; //Valores de 0 a 99
  Wire.beginTransmission(DS1307_ADDRESS);
  Wire.write(zero); //Stop no CI para que o mesmo possa receber os dados

  //As linhas abaixo escrevem no CI os valores de
  //data e hora que foram colocados nas variaveis acima
  Wire.write(ConverteParaBCD(segundos));
  Wire.write(ConverteParaBCD(minutos));
  Wire.write(ConverteParaBCD(horas));
  Wire.write(ConverteParaBCD(diadasemana));
  Wire.write(ConverteParaBCD(diadomes));
  Wire.write(ConverteParaBCD(mes));
  Wire.write(ConverteParaBCD(ano));
  Wire.write(zero); //Start no CI
  Wire.endTransmission();
}
```

Figuras 22, 23, 24 - Código Arduino

Outro foco dessa pesquisa se dá na área da economia. O protótipo do sistema teve um custo consideravelmente menor do que os Data Loggers encontrados atualmente no mercado (Figura 25).

Item	Preço
Sensor LDR	R$ 1,00
Sensor LM35	R$ 4,00
Sensor DHT11	R$ 6,00
Potenciômetro 10kΩ	R$ 1,50
CI DS1307	R$ 4,00
Bateria Lítio 3V	R$ 6,00
Cristal 32.768 kHz	R$ 3,00

Display LCD 16x2	R$ 12,00
Módulo *SD Card*	R$ 9,00
Cartão de memória 4GB	R$ 13,00
30 Jumper's	R$ 9,00
Arduino Mega 2560 R3	R$ 60,00
Módulo *Ethernet SHIELD* W5100	R$ 50,00
Total	**R$ 178,50**

Figura 25 – Tabela de preços

Durante as etapas anteriores, o sistema será alimentado por meio da rede elétrica, abordagem que visa isolar problemas de alimentação. Assim que o sistema for construído por completo, iniciar-se-á a pesquisa por fontes de alimentação renováveis. Pretende-se utilizar placas de energia solar de baixo custo [5] para alimentar o sistema; então, qualquer *hardware* adicional que necessitar ser confeccionado, será projetado e construído nesta etapa.

Após a construção do módulo de alimentação, o sistema será submetido a um teste de regressão a fim de verificar se os dados armazenados possuem a mesma qualidade ao serem comparados aos dados obtidos quando o sistema está conectado na rede elétrica. Assim que tudo estiver integrado e testado, será desenvolvida documentação para garantir que o projeto possa ser replicado por grupos de pesquisas que necessitem utilizá-lo.

5 CONSIDERAÇÕES FINAIS

Com o desenvolvimento do protótipo descobrimos que era possível realizar a construção de um *Data Logger* que atenda às exigências inicialmente determinadas como base para a execução deste projeto, onde foram alcançadas as metas de preço e de eficiência.

Com o módulo de *SD Card* para Arduino encontramos uma maneira eficiente e barata para o armazenamento dos dados, que é uma das principais características de um *Data Logger*. Obtivemos sucesso na leitura dos sensores, na exibição dos valores dos mesmos em um display LCD e no funcionamento das placas de circuito impresso.

A grande dificuldade encontrada foi na apresentação dos dados lidos de forma remota com a utilização do Módulo *Ethernet Shield* W5100, onde utilizamos técnicas aprendidas fora da ementa do curso.

O protótipo ainda receberá alimentação por meio de placas solares, assim concluindo o que foi proposto inicialmente para execução do projeto.

REFERÊNCIAS

1. Zhang, C., and J. Zhang. "**Current Techniques for Detecting and Monitoring Algal Toxins and Causative Harmful Algal Blooms**." *J Environ Anal Chem* 2.123 (2015): 2.

2. Catalogo de produtos da Novus. Disponível em <http://www.novus.com.br/site/default.asp?TroncoID=621808&SecaoID=803220&SubsecaoID=0>.

3. GUANZIROLI, C. E.; CARDIM, S. E. C. S. Novo Retrato da Agricultura Familiar: O Brasil Redescoberto. **Projeto de Cooperação Técnica INCRA / FAO: Ministério do Desenvolvimento Agrário.** Brasília, 2000.

4. Revista Galileu. **Campo Hackeado**. Disponível em <http://revistagalileu.globo.com/Revista/noticia/2014/03/campo-hackeado.html>. Data de acesso: 19 ago. 2015.

5. Beyond solar radiation management – the strategic role of low-cost photovoltaics in solar energy production. DOI:10.1080/14786451.2013.854244 Felix Hermerschmidt[a], Panayiotis D. Pouloupatis[a], George Partasides[b], Andreas Lizides[b], Stella Hadjiyiannakou[b] & Stelios A. Choulis[a*] pages 211-220

6. Testo 176 P1 - *Data logger* Pressão, temperatura e humidade. **Testo**. Disponível em: <https://www.testo.com.br/detalhes_do_produto/0572+1767/testo-176-P1-Data-loggerPressao-temperatura-e-humidade#tab-8>. Acesso em: 20 ago. 2015.

7. ARDUINO UNO. Site oficial do Arduino UNO. Disponível em: <http://www.arduino.cc/en/Main/ArduinoBoardUno>. Acesso em: 20 ago. 2015.

8. YARI, J. UNIVERSIDADE ANHANGUERA – UNIDERP Programa de Mestrado Profissional em Produção e Gestão Agroindústria. **Desenvolvimento de Miniplataforma de Coleta de Dados Meteorológicos para Pequenos Produtores Rurais Utilizando as Tecnologias Livres Arduino e Android.** Campo Grande - MS, 2013. Disponível em: <https://s3.amazonaws.com/pgsskrotondissertacoes/4eb662f38545dafa08ddf3988dc7947c.pdf>. Acesso em: 21 ago. 2015.

9. INTRUCTABLES. **Using the ESP8266 module.** Disponível em: <http://www.instructables.com/id/Using-the-ESP8266-module/>. Acesso em: 22 ago. 2015.

10. LIMA. DANIEL MARINO PEREIRA. Universidade Federal do Paraná. **Sensor de Temperatura - Modelo lm35 – National.** Disponível em: <http://www.eletrica.ufpr.br/edu/Sensores/20071/LM35.html>. Acesso em: 22 ago. 2015.

11. AOSONG. **Temperature and Humidity Module – DHT11 Product Manual.** Disponível em: <http://akizukidenshi.com/download/ds/aosong/DHT11.pdf>. Acesso em: 23 ago. 2015.

12. SCRIBD.. **Artigo LDR x Sensor de Luminosidade _teste3_rev4**, 2014. Disponivel em: <http://pt.scribd.com/doc/219049351/Artigo-LDR-x-Sensor-de- Luminosidade-teste3-rev4#scribd>. Acesso em: 24 ago. 2015.

13. CADSOFT COMPUTER INC. Disponível em: <http://www.cadsoftusa.com>. Acesso em: 24 ago. 2015.