CAPTURA DE TEMPERATURA E UMIDADE DE DATA CENTERS

TEMPERATURE AND HUMIDITY SENSOR WITH ARDUINO FOR A DATA CENTER

Lucas Grahl de Oliveira
Graduando em Rede de Computadores pela Fatec Bauru
E-mail: lugrahl14@gmail.com

Paulo Roberto Anzolin Filho
Graduando em Rede de Computadores pela Fatec Bauru
E-mail: pauloanzolin2304@gmail.com

Prof.° Gustavo Cesar Bruschi
Docente na Fatec Bauru
E-mail: gustavo.bruschi@fatec.sp.gov.br

FATEC - BAURU

RESUMO:

Um risco existente atualmente nos ambientes de datacenter é a falta de controle de temperatura e umidade de uma sala rack feito de forma eficaz. Problemas como curtos-circuitos e incêndios não são impossíveis, principalmente dentro de Data Center que funcionam de maneira ininterrupta e os equipamentos podem chegar a 60°C sem o devido controle ou refrigeração. Com a grande quantidade de equipamentos nos Data Center é importante manter uma temperatura estável. O equipamento em desenvolvimento proporciona baixo consumo de energia e alta precisão nos valores. Este artigo tem como proposta apresentar um dispositivo mais barato e confiável para a captura de temperatura e umidade com o Arduino trabalhando em conjunto com os módulos LoRa (Long Range), DHT11 (Capturador de temperatura e umidade), Arduino UNO e Dragino Gateway para envio dos dados para a internet. Para este fim o Arduino em conjunto ao módulo DTH11 será capaz de coletar as informações do ambiente e transmitir os dados pelo LoRa a fim de evitar atrasos nas informações. Para o desenvolvimento do projeto, foi criado um protótipo que é testado em sala rack. O teste foi realizado com diferentes temperaturas junto a um termo-higrômetro para validar os resultados. O resultado obtido atendeu as expectativas, a programação do equipamento funcionou perfeitamente, obtendo capturas a cada 20 segundos por tempo ilimitado e assim fazendo a comunicação necessária para o tratamento dos dados com a nuvem utilizando o Dragino Gateway. Conclui-se que, mesmo com um equipamento mais barato é possível obter a mesma confiabilidade nos resultados, comparando com equipamentos de nível industrial.

Palavras-chave: Arduino; eficiência energética; Iot; tecnologia.

ABSTRACT:

A risk currently existing in data center environments is the lack of effective temperature control of a rack room. Problems such as short circuits and fires are not impossible, especially in data centers that operate uninterruptedly and equipment can reach 60°C without proper control or cooling. With the large amount of equipment in Data Centers it is important to maintain a stable temperature. The equipment under development provides low power consumption and high accuracy in the values. This article proposes to present a cheaper and more reliable device to capture temperature and humidity with the Arduino working together with the modules LoRa (Long Range), DHT11 (Temperature and Humidity Capture), Arduino UNO and Dragino Gateway to send the data to the internet. For this purpose, the Arduino together with the DTH11 module will be able to collect information from the environment and transmit the data through LoRa in order to avoid delays in the information. For the development of the project, a prototype was created and tested in a rack room. The test was performed with different temperatures together with a thermo-hygrometer to validate the results. The result obtained met expectations, the programming of the equipment worked perfectly, obtaining captures every 20 seconds for unlimited time and thus making the necessary communication for the treatment of data with the cloud using Dragino Gateway. It is concluded that, even with a cheaper equipment it is possible to obtain the same reliability in the results, compared with industrial level equipment.

Key-Words: Arduino; energy efficincy; Iot; Tecnology.

1 INTRODUÇÃO

Segundo a IBM a gestão de risco no TI é de extrema importância atualmente pois tal setor vem se tornando cada vez mais o principal fator de muitas empresas, é muito importante na gestão de riscos planos para mitigação e prevenção total de riscos. Um risco existente hoje em dia é a falta de controle de temperatura de uma sala rack feito de forma eficaz, a diferença de 4° Celsius dentro pode significar perda total de um equipamento ou o funcionamento correto do mesmo. Problemas como curtos-circuitos e incêndios não são impossíveis, principalmente dentro de Data Center que funcionam de maneira ininterrupta, a temperatura pode chegar a 60°C. Problemas com umidade também podem acontecer, a variação constante de temperatura junto à alta umidade podem causar pontos de orvalhos podendo gerar curto ou até mesmo oxidar os dispositivos. Para evitar que tais acidentes ocorram foram criadas diretrizes térmicas desenvolvidas pela Sociedade Americana de Engenheiros de Aquecimento, Refrigeração e Ar-Condicionado (ASHRAE) tais diretrizes ditam que a umidade relativa do ar deve ser mantida entre 20% e 80% e a temperatura do ambiente deve ser entre 15°C e 32°C.

O Arduíno é um dos equipamentos mais utilizados hoje em dia para desenvolver e criar pequenos gadgets, desde os mais simples aos mais sofisticados. Como exemplo, um sensor de pressão e mapear uma sala 3d para um programa, sua demanda vem crescendo devido a grande quantidade de módulos que podem ser acrescentados ao dispositivo, existindo uma grande lista de programas que podem ser desenvolvidos com inúmeras combinações desses módulos. Em busca por redução de custos, mas não a perda de qualidade e eficiência, estimulou o uso do dispositivo que por sua vez não dispensa qualidade, ainda mais com o consumo de energia reduzido, precisa de apenas 5 volts para alimentar o dispositivo com poucos módulos acrescentados a ele.

O objetivo deste projeto é criar um dispositivo que seja capaz de coletar dados como temperatura e umidade do ambiente de forma eficaz e eficiente, tal dispositivo também deve ser de baixo custo. Podendo ser considerado um termo-higrômetro, o Arduino com DHT11 e conjunto com um software IoT é capaz de gerar relatórios e gráficos podendo também gerar alertas por e-mail ou pop-up no servidor se necessário. Atualmente, os sensores existentes possuem um custo muito alto tanto para compra inicial quanto para manutenção.

2 FUNDAMENTAÇÃO TEÓRICA

O projeto é fundamentado na necessidade de controlar a temperatura e a umidade de uma sala rack a qual possui dispositivos sensíveis e de extrema importância para a empresa. Utilizando um dispositivo Arduino com LoRa (Long Range) e sensores de temperatura e umidade (DHT11), é possível coletar as informações necessárias e transmitir para qualquer outro dispositivo fora da sala via ondas WiFi ou Radio. Estes módulos são conectados ao Arduíno e as informações são transmitidas para o Dragino Gateway que faz a tratativa das informações enviando os dados para um servidor Iot onde é mostrado em gráfico todas as informações coletadas.

2.1 Arduino

Segundo o Arduíno (2020), Arduino é uma plataforma open source ou hardware para prototipagem eletrônica, projetada com um microcontrolador Atmel AVR com suporte para entrada/saída dados já embutido, com linguagem de programação padrão baseado no em C/C++, mas explicando de uma forma bem simples.

Utilizamos Arduino hoje porque por causa da velocidade que se consegue desenvolver algum protótipo ou até mesmo um produto, e se é rápido também é mais simples, isso é conseguido devido a modularidade do Arduino, da farta documentação e da grande quantidade de módulos disponíveis para se conectar ao Arduino, como sensores de luz, distância, alternadores, motores, etc.

O software Arduino (IDE) roda em sistemas Windows, Mac OSX e distribuições Linux, sendo muitos microcontroladores limitados para o Windows, possibilitando então que os equipamentos sejam programados a partir de qualquer sistema citado anteriormente. O software é de fácil uso para iniciantes, é flexível para que usuários mais avançados possam utilizá-lo.

2.2 DHT11 – Sensor de temperatura e umidade

De acordo com Blogeletrogate (2020), o eletrônico é um equipamento duplo, podendo captar a temperatura e umidade. A captação de umidade sendo um sensor capacitivo e o de temperatura um sensor termistor NTC, que é um resistor sensível à temperatura. O dispositivo tem a capacidade de transmitir os dados já de forma digital na porta de saída por meio de um microcontrolador.

2.3 LoRa Shield – Long Range

Segundo Teixeira (2017), LoRa é uma tecnologia de modulação de espectro, ou seja, um dispositivo de comunicação sem fio que funciona em baixas frequências de rádio. Capaz de realizar transmissões em longa distâncias com o mínimo de energia, seu alcance em áreas urbanas é de 4 Km e áreas urbanas pode chegar a 12 Km, porém a Anatel (Agência Nacional de Telecomunicações) limitou a potência para um raio de 2 Km para alguns modelos.

2.4 Dragino - Gateway

Segundo Dragino Technology Co., LTD. (2021) Dragino é um projeto OpenWrt de baixo custo, com código e hardware aberto, possuindo suporte total a Ethernet ou conexão WiFi 802.11b/g/n, que tem como função resolver e melhorar a conexão de microcontroladores. É possível utilizá-lo para controle remoto, coleta de dados, aplicação web, conexão mesh, etc.

2.5 ThingSpeak

De acordo com The MathWorks, Inc. ThingSpeak é uma plataforma de análise IoT que permite agregar, visualizar e analisar dados em tempo real pela nuvem. É possível enviar dados para o ThingSpeak de qualquer dispositivo que conectam a internet criando automaticamente gráficos com os dados e permitindo a fácil criação e envio de alertas por serviços web, existindo tanto na versão web quanto aplicativo mobile para a visualização destes gráficos.

2.6 Data Center

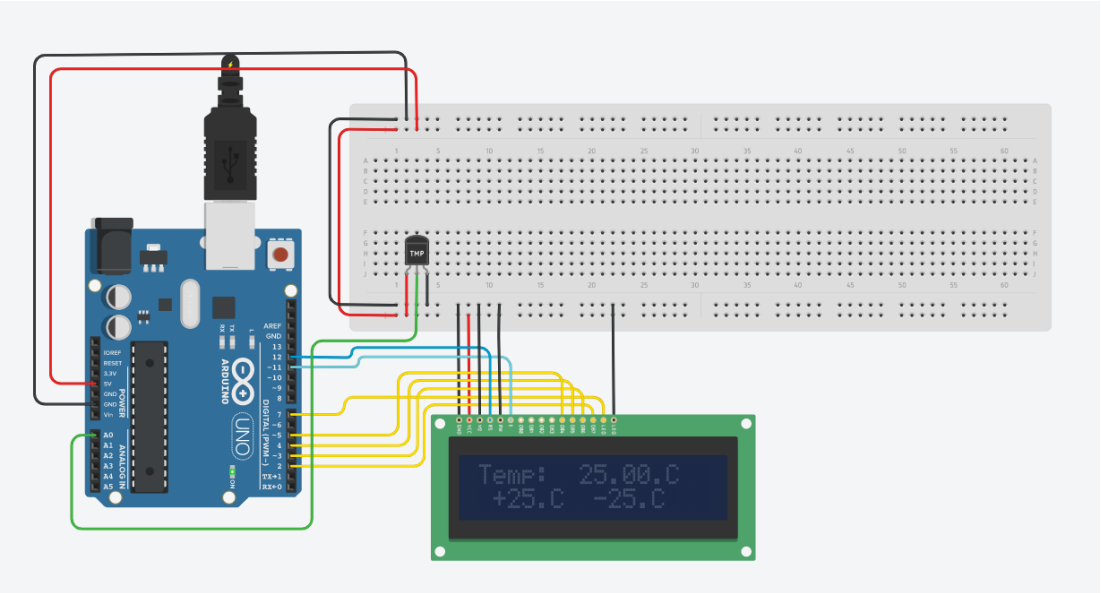
Concordando com a Cisco (2021), data center é onde se concentra toda a Infraestrutura de Redes de uma empresa, temos nela servidores (AD, Arquivos, DNS, GPOs) Nobreaks, Firewall, Load Balance, entre outros recursos necessário para que a rede funcione de acordo com a necessidade. Essas também podem estar em nuvem, um servidor não instalado localmente, mas em servidores do provedor contratante.

Os dados que existem estão conectados entre esses data centers, tanto em redes públicas ou privadas. Essas redes podem-se comparar com uma teia de aranha, onde cada cruzamento pode ser chamado de data center. Quando utilizado algum aplicativo hospedado na nuvem, eles estão utilizando data centers e esse provedor se encontra em nuvem.

3 MATERIAIS E MÉTODOS

Com o objetivo de aumentar a segurança física do data center, sem perder a confiança e controle foi sugerido a implementação de um protótipo de um controlador com Arduino como mostrado abaixo na figura 1. O controlador tem como objetivo coletar a temperatura e a umidade do ambiente e registrar em um banco de dados web para uma fácil consulta de qualquer lugar e de qualquer dispositivo.

Figura 1 – Circuito Simulado com TMP36 e Display LCD 16x2



Fonte: Os Autores (2021)

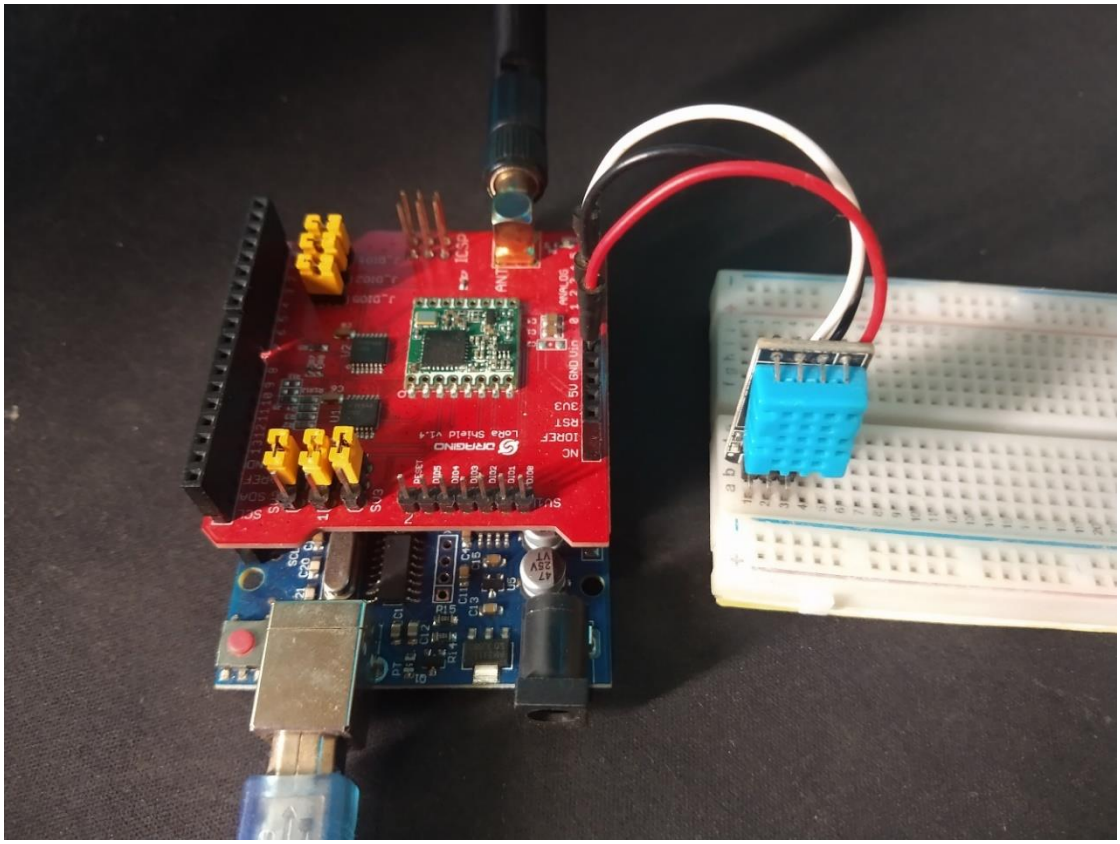
O intuito do projeto é aproveitar algo já existente e conhecido no mercado. O circuito é bastante simples, mas de alto fator de customização, contendo apenas uma alimentação de 5v, uma placa Arduíno e um sensor DHT11. No exemplo protótipo foi usado um módulo de temperatura TMP36 muito similar ao DHT11, porém apenas obtém informações da temperatura do ambiente, e um display 16x2 para exibição das informações, a configuração do protótipo é feita via software Tikercad um site que simula a conexão desses eletrônicos, assim como sua configuração. O Autodesk Tinkercad é uma coletânea de ferramentas online e grátis que permite criar essas simulações.

Diferente do protótipo no projeto foi utilizado um Arduino Uno, composto pelos módulos LoRa Shield e DHT11 - mostrados na figura 2 - comunicando com o Dragino Gateway, figura 3. Todos os dados coletados por sua vez são transmitidos para o ThingSpeak, um banco de dados para dispositivo IoT possuindo layout para visualização web como mostrado na figura 4 e um layout para dispositivo mobile, figura 5.

É possível acessar tais informações de qualquer local e de um dispositivo com acesso à internet. Para definir a configuração utilizamos o software Arduino IDE e com ele programamos o Arduíno UNO e o Dragino Gateway para que os módulos possam receber as informações captadas do módulo DHT11, transmitir do Módulo LoTa Shield para o Dragino Gateway e por fim enviar via internet para o banco de dados IoT.

Para que a comunicação funcione utilizamos o código para que os componentes da figura 2 se comuniquem com o Dragino Gateway da figura 3, o mesmo deve acontecer ao inverso, no quadro 2 temos o código utilizado da comunicação tanto com o Arduíno e a chave utilizada para a comunicação com o banco de dados ThingSpeak.

Figura 2 – Arduíno Uno, LoRa Shield e DHT11



Fonte: Os Autores (2021)

Figura 3 – Dragino Gateway

Fonte: Os Autores (2021)

Quadro 1 – Código utilizado no Arduino Uno

```
#include <SPI.h>
#include <RH_RF95.h>

RH_RF95 rf95;

#define dht_dpin A0 // Use A0 pin as Data pin for DHT11.
byte bGlobalErr;
char dht_dat[5]; // Store Sensor Data
char node_id[3] = {1,1,1}; //LoRa End Node ID
```

```
float frequency = 868.0;
unsigned int count = 1;

void setup()
{
InitDHT();
Serial.begin(9600);
if (!rf95.init())
Serial.println("init failed");
// Setup ISM frequency
rf95.setFrequency(frequency);
// Setup Power,dBm
rf95.setTxPower(13);
rf95.setSyncWord(0x34);

Serial.println("LoRa End Node Example --");
Serial.println("    DHT11 Temperature and Humidity Sensor\n");
Serial.print("LoRa End Node ID: ");

for(int i = 0;i < 3; i++)
{
Serial.print(node_id[i],HEX);
}
Serial.println();
}

void InitDHT()
{
pinMode(dht_dpin,OUTPUT);//Set A0 to output
digitalWrite(dht_dpin,HIGH);//Pull high A0
}

//Get Sensor Data
void ReadDHT()
{
```

```
bGlobalErr=0;
byte dht_in;
byte i;

//pinMode(dht_dpin,OUTPUT);
digitalWrite(dht_dpin,LOW);//Pull Low A0 and send signal
delay(30);//Delay > 18ms so DHT11 can get the start signal

digitalWrite(dht_dpin,HIGH);
delayMicroseconds(40);//Check the high level time to see if the data is 0 or 1
pinMode(dht_dpin,INPUT);
// delayMicroseconds(40);
dht_in=digitalRead(dht_dpin);//Get A0 Status
//   Serial.println(dht_in,DEC);
if(dht_in){
bGlobalErr=1;
return;
}
delayMicroseconds(80);//DHT11 send response, pull low A0 80us
dht_in=digitalRead(dht_dpin);

if(!dht_in){
bGlobalErr=2;
return;
}
delayMicroseconds(80);//DHT11 send response, pull low A0 80us
for (i=0; i<5; i++)//Get sensor data
dht_dat[i] = read_dht_dat();
pinMode(dht_dpin,OUTPUT);
digitalWrite(dht_dpin,HIGH);//release signal and wait for next signal
byte dht_check_sum = dht_dat[0]+dht_dat[1]+dht_dat[2]+dht_dat[3];//calculate check sum
if(dht_dat[4]!= dht_check_sum)//check sum mismatch
{bGlobalErr=3;}
};
```

```
byte read_dht_dat(){
byte i = 0;
byte result=0;
for(i=0; i< 8; i++)
{
while(digitalRead(dht_dpin)==LOW);//wait 50us
delayMicroseconds(30);//Check the high level time to see if the data is 0 or 1
if (digitalRead(dht_dpin)==HIGH)
result |=(1<<(7-i));//
while (digitalRead(dht_dpin)==HIGH);//Get High, Wait for next data sampleing.
}
return result;
}
uint16_t calcByte(uint16_t crc, uint8_t b)
{
uint32_t i;
crc = crc ^ (uint32_t)b << 8;

for ( i = 0; i < 8; i++)
{
if ((crc & 0x8000) == 0x8000)
crc = crc << 1 ^ 0x1021;
else
crc = crc << 1;
}
return crc & 0xffff;
}

uint16_t CRC16(uint8_t *pBuffer,uint32_t length)
{
uint16_t wCRC16=0;
uint32_t i;
if (( pBuffer==0 )||( length==0 ))
{
return 0;
```

```
}
for ( i = 0; i < length; i++)
{
wCRC16 = calcByte(wCRC16, pBuffer[i]);
}
return wCRC16;
}

void loop()
{
Serial.print("############    ");
Serial.print("COUNT=");
Serial.print(count);
Serial.println("    ############");
count++;
ReadDHT();
char data[50] = {0} ;
int dataLength = 7; // Payload Length
// Use data[0], data[1],data[2] as Node ID
data[0] = node_id[0] ;
data[1] = node_id[1] ;
data[2] = node_id[2] ;
data[3] = dht_dat[0];//Get Humidity Integer Part
data[4] = dht_dat[1];//Get Humidity Decimal Part
data[5] = dht_dat[2];//Get Temperature Integer Part
data[6] = dht_dat[3];//Get Temperature Decimal Part
switch (bGlobalErr)
{
case 0:
Serial.print("Current humidity = ");
Serial.print(data[3], DEC);//Show humidity
Serial.print(".");
Serial.print(data[4], DEC);//Show humidity
Serial.print("%  ");
Serial.print("temperature = ");
```

```
Serial.print(data[5], DEC);//Show temperature
Serial.print(".");
Serial.print(data[6], DEC);//Show temperature
Serial.println("C  ");
break;
case 1:
Serial.println("Error 1: DHT start condition 1 not met.");
break;
case 2:
Serial.println("Error 2: DHT start condition 2 not met.");
break;
case 3:
Serial.println("Error 3: DHT checksum error.");
break;
default:
Serial.println("Error: Unrecognized code encountered.");
break;
}

uint16_t crcData = CRC16((unsigned char*)data,dataLength);//get CRC DATA
//Serial.println(crcData,HEX);

Serial.print("Data to be sent(without CRC): ");

int i;
for(i = 0;i < dataLength; i++)
{
Serial.print(data[i],HEX);
Serial.print(" ");
}
Serial.println();

unsigned char sendBuf[50]={0};

for(i = 0;i < dataLength;i++)
```

```
{
sendBuf[i] = data[i] ;
}

sendBuf[dataLength] = (unsigned char)crcData; // Add CRC to LoRa Data
sendBuf[dataLength+1] = (unsigned char)(crcData>>8); // Add CRC to LoRa Data

Serial.print("Data to be sent(with CRC):    ");
for(i = 0;i < (dataLength +2); i++)
{
Serial.print(sendBuf[i],HEX);
Serial.print(" ");
}
Serial.println();

rf95.send(sendBuf, dataLength+2);//Send LoRa Data

uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];//Reply data array
uint8_t len = sizeof(buf);//reply data length

if (rf95.waitAvailableTimeout(3000))// Check If there is reply in 3 seconds.
{
// Should be a reply message for us now
if (rf95.recv(buf, &len))//check if reply message is correct
{
if(buf[0] == node_id[0] && buf[1] == node_id[2] && buf[2] == node_id[2] ) // Check if reply m
essage has the our node ID
{
pinMode(4, OUTPUT);
digitalWrite(4, HIGH);
Serial.print("Got Reply from Gateway: ");//print reply
Serial.println((char*)buf);

delay(400);
digitalWrite(4, LOW);
```

```
//Serial.print("RSSI: ");  // print RSSI
//Serial.println(rf95.lastRssi(), DEC);
}
}
else
{
Serial.println("recv failed");//
rf95.send(sendBuf, strlen((char*)sendBuf));//resend if no reply
}
}
else
{
Serial.println("No reply, is LoRa gateway running?");//No signal reply
rf95.send(sendBuf, strlen((char*)sendBuf));//resend data
}
delay(3000); // Mandar dados a cada 3 segundos
Serial.println("");
}
```

Fonte: Os Autores (2021)

Quadro 2 – Código Dragino Gateway

```
#include <SPI.h>
#include <RH_RF95.h>
#include <Console.h>
#include <Process.h>
RH_RF95 rf95;

//If you use Dragino IoT Mesh Firmware, uncomment below lines.
//For product: LG01.
#define BAUDRATE 115200

String myWriteAPIString = " chave de leitura ";
uint16_t crcdata = 0;
uint16_t recCRCData = 0;
float frequency = 868.0;
String dataString = "";

void uploadData(); // Upload Data to ThingSpeak.

void setup()
{
   Bridge.begin(BAUDRATE);
   Console.begin();
   // while(!Console);
   if (!rf95.init())
      Console.println("init failed");
   ;
   // Setup ISM frequency
   rf95.setFrequency(frequency);
   // Setup Power,dBm
   rf95.setTxPower(13);
   rf95.setSyncWord(0x34);
```

```
    Console.println("LoRa Gateway Example  --");
    Console.println("    Upload Single Data to ThinkSpeak");
}

uint16_t calcByte(uint16_t crc, uint8_t b)
{
    uint32_t i;
    crc = crc ^ (uint32_t)b << 8;

    for ( i = 0; i < 8; i++)
    {
      if ((crc & 0x8000) == 0x8000)
        crc = crc << 1 ^ 0x1021;
      else
        crc = crc << 1;
    }
    return crc & 0xffff;
}

uint16_t CRC16(uint8_t *pBuffer, uint32_t length)
{
    uint16_t wCRC16 = 0;
    uint32_t i;
    if (( pBuffer == 0 ) || ( length == 0 ))
    {
        return 0;
    }
    for ( i = 0; i < length; i++)
    {
        wCRC16 = calcByte(wCRC16, pBuffer[i]);
    }
    return wCRC16;
}

uint16_t recdata( unsigned char* recbuf, int Length)
```

```
{
   crcdata = CRC16(recbuf, Length - 2); //Get CRC code
   recCRCData = recbuf[Length - 1]; //Calculate CRC Data
   recCRCData = recCRCData << 8; //
   recCRCData |= recbuf[Length - 2];
}
void loop()
{
   if (rf95.waitAvailableTimeout(2000))// Listen Data from LoRa Node
   {
      uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];//receive data buffer
      uint8_t len = sizeof(buf);//data buffer length
      if (rf95.recv(buf, &len))//Check if there is incoming data
      {
         recdata( buf, len);
         Console.print("Get LoRa Packet: ");
         for (int i = 0; i < len; i++)
         {
            Console.print(buf[i],HEX);
            Console.print(" ");
         }
         Console.println();
         if(crcdata == recCRCData) //Check if CRC is correct
         {
            if(buf[0] == 1 && buf[1] == 1 && buf[2] ==1) //Check if the ID match the LoRa No
de ID
            {
               uint8_t data[] = "   Server ACK";//Reply
               data[0] = buf[0];
               data[1] = buf[1];
               data[2] = buf[2];
               rf95.send(data, sizeof(data));// Send Reply to LoRa Node
               rf95.waitPacketSent();
               int newData[4] = {0, 0, 0, 0}; //Store Sensor Data here
               for (int i = 0; i < 4; i++)
```

```
            {
                newData[i] = buf[i + 3];
            }
            int hh = newData[0];
            int hl = newData[1];
            int th = newData[2];
            int tl = newData[3];
            Console.print("Get Temperature:");
            Console.print(th);
            Console.print(".");
            Console.println(tl);
            Console.print("Get Humidity:");
            Console.print(hh);
            Console.print(".");
            Console.println(hl);

            dataString ="field1=";
            dataString += th;
            dataString +=".";
            dataString += tl;
            dataString +="&field2=";
            dataString += hh;
            dataString +=".";
            dataString += hl;

            uploadData(); //
            dataString="";
        }
    }
    else
      Console.println(" CRC Fail");
}
else
{
    //Console.println("recv failed");
```

```
            ;
        }
    }

}

void uploadData() {//Upload Data to ThingSpeak
  // form the string for the API header parameter:

  // form the string for the URL parameter, be careful about the required "
  String upload_url = "https://api.thingspeak.com/update?api_key=";
  upload_url += myWriteAPIString;
  upload_url += "&";
  upload_url += dataString;

  Console.println("Call Linux Command to Send Data");
  Process p;    // Create a process and call it "p", this process will execute a Linux curl com
mand
  p.begin("curl");
  p.addParameter("-k");
  p.addParameter(upload_url);
  p.run();    // Run the process and wait for its termination

  Console.print("Feedback from Linux: ");
  // If there's output from Linux,
  // send it out the Console:
  while (p.available()>0)
  {
    char c = p.read();
    Console.write(c);
  }
  Console.println("");
  Console.println("Call Finished");
  Console.println("########################################");
  Console.println("");
```

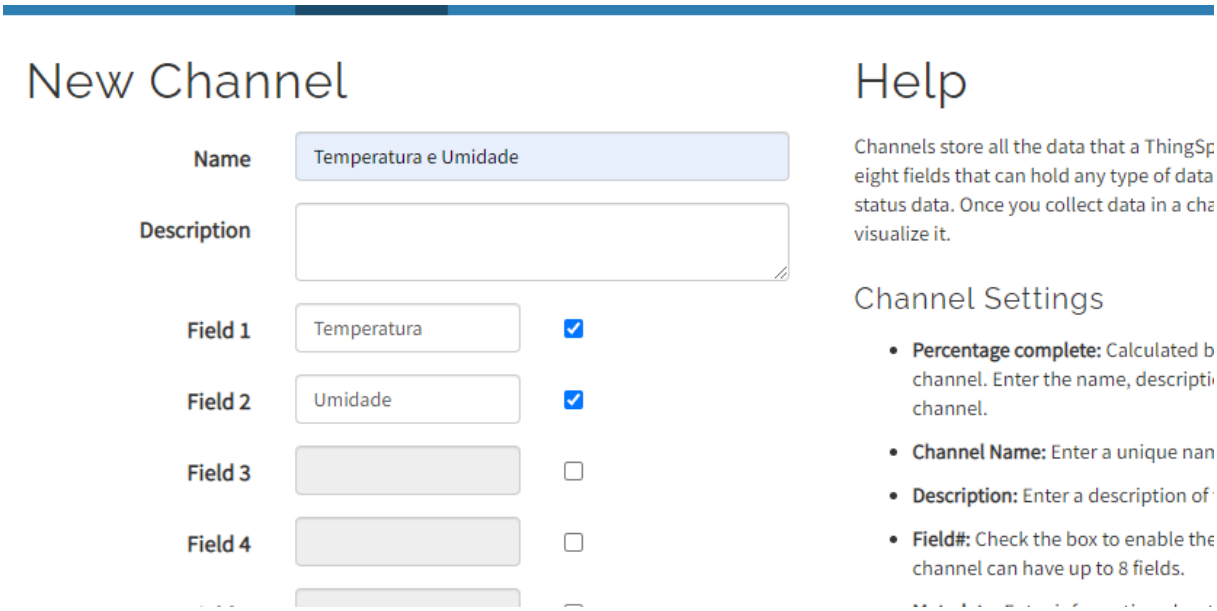
```
}
```

Fonte: Os Autores (2021)

Para o funcionamento do banco de dados IoT ThingSpreak, é necessário que se crie uma chave de comunicação, para que o Gateway Dragino possa enviar os dados diretamente para as tabelas criadas np bando de dados IoT, como mostrado abaixo na figura 4.

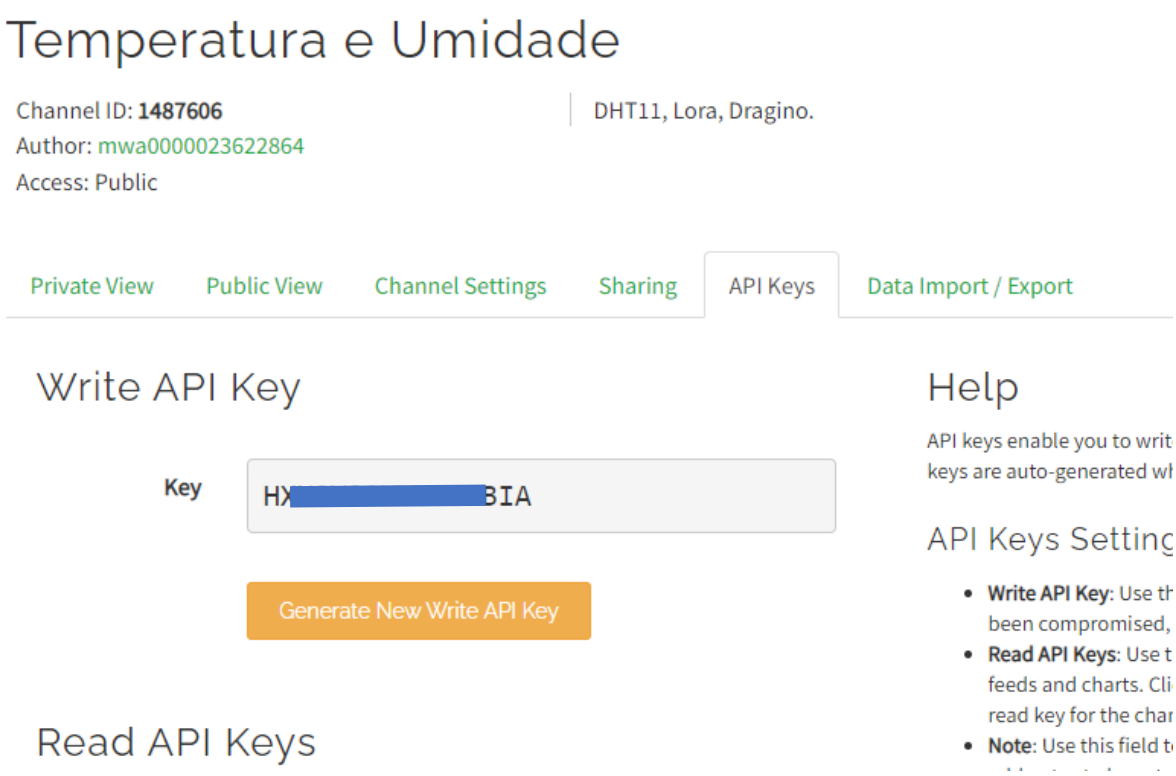
Depois de criada as tabelas o ThingSpeak irá gerar uma chave de comunicação mostrado na figura 5; para que seja inserido no código do Dragino Gateway do quadro 2, na linha do código onde descrito “String myWriteAPIString = "chave de leitura";”, é substituído a “chave de leitura” pela cheve gerada pelo StingSpeak. Depois de configurado e os dispositivos ligados, as tabelas iram começar a receber os valores da temperatura e umidade como mostrado na figura 6 e figura 7.

Figura 4 – ThingSpeak Web - Tabelas



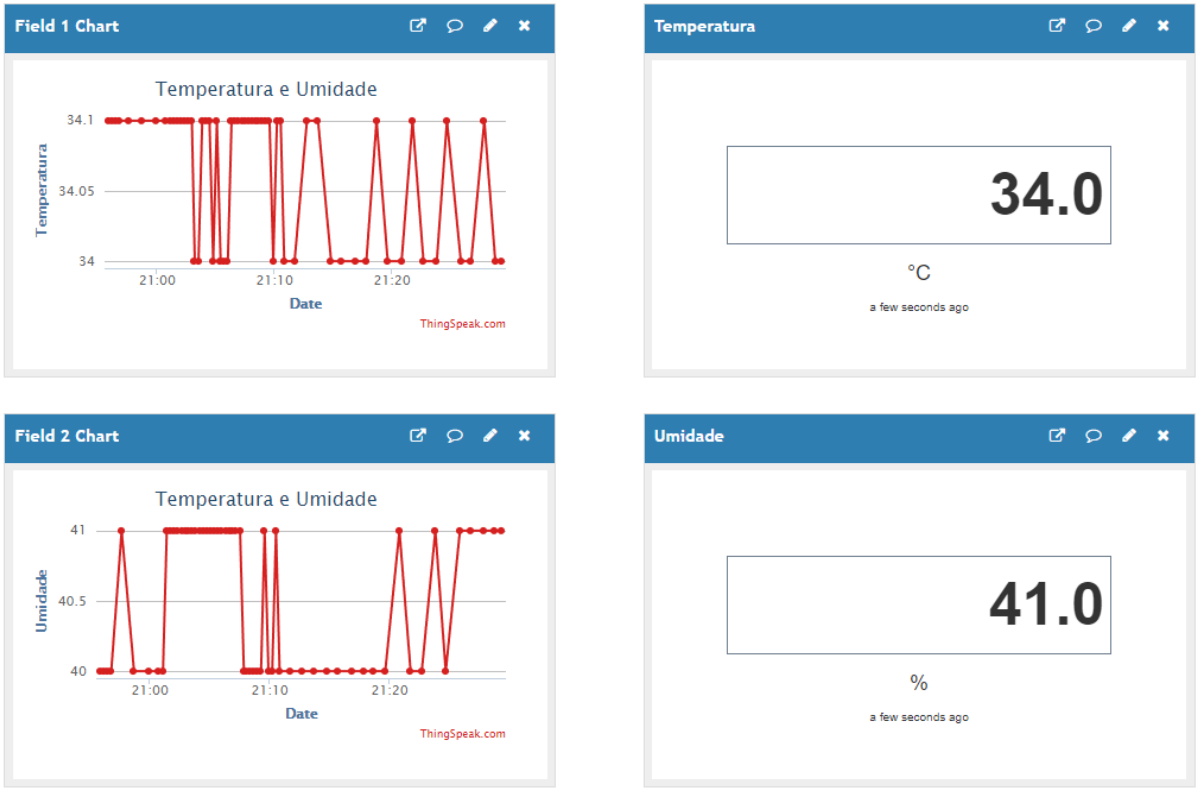
Fonte: https://thingspeak.com/channels (2021)

Figura 5 – ThingSpeak Web (Chave)



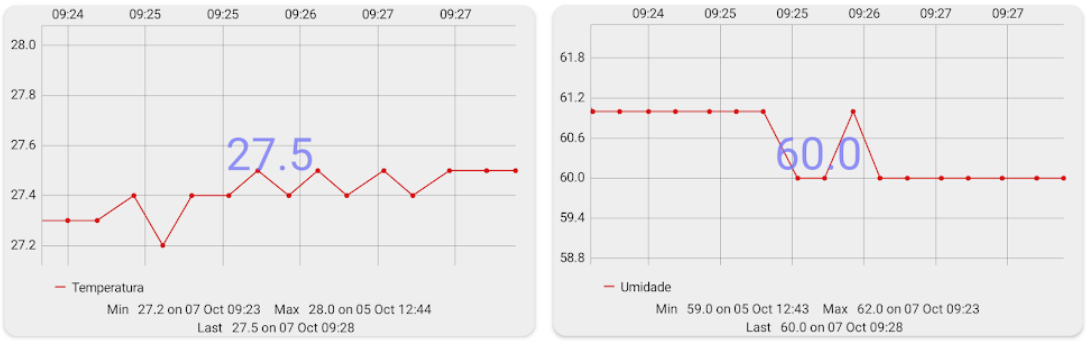
Fonte: https://thingspeak.com/channels (2021)

Figura 6 – ThingSpeak Web - Gráficos



Fonte: https://thingspeak.com/ (2021)

Figura 7 – ThingSpeak Mobile



Fonte: Os Autores (2021)

4 RESULTADOS OBTIDOS

Diante da ideia proposta, o dispositivo foi instalado nos data center para teste e com isso forneceu sem dificuldades os dados de temperatura e umidade, fazendo uma conexão via rádio, utilizando o LoRa, para o coletor principal e posteriormente enviando para um servidor WEB foi exibido todas as informações. Os valores são atualizados em média a cada 15 segundos com o equipamento numa distância de até 700m dentro do limite urbano, devido a utilização do LoRa Shield. Deixamos o equipamento ligado por 14 dias seguidos em um único ambiente para o controle e obtenção de dados. Não houve falhas em sua utilização ou grande variância nos resultados.

O LoRa necessita de 90mA (TX) de alimentação em sua fase mais ativa, em sua fase menos ativa utiliza apenas 220uA em modo Sleeping, utilizando a tecnologia wireless RS232 / RS485, uma frequência de até 915Mhz, com essas especificações é possível chegar a uma distância de comunicação de até 4km em áreas urbanas, obtivemos resultados próximos ao realizar testes de transmissão de informação mas foi optado por limitar a 700m afim de facilitar o controle e evitar a perca de pacotes, esses valores podem ser afetados de acordo com a fabricante e condição de instalação. Os códigos utilizados são baseados em C# utilizando bibliotecas de terceiros para que os módulos pudessem comunicar entre si, os códigos são open source o que permite fazer alteração se necessário, inclusive acrescentar outros módulos de acordo com a necessidade e preferência.

5 CONCLUSÃO

Cada vez mais a segurança de salas racks em relação a temperatura vem se tornando uma das principais preocupações, deixando apenas de ser algo complementar e se tornando uma prioridade principal. A aplicação em residência e indústria vem se tornando cada vez mais comum com a redução dos preços e custo de energia elétrica. Além do conforto físico para as pessoas a redução da temperatura de um ambiente pode ser benéfica também para dispositivos que aquecem naturalmente por estarem ativados e possuírem alto nível de trabalho que produza calor de alguma maneira é nesse ponto que o controle de temperatura se torna importante, conta com a sensação térmica do corpo humano é ineficaz já que uma máquina trabalha com valores bem abaixo.

Este projeto apresentou um dispositivo de controle de temperatura e umidade com o objetivo de facilitar o controle e reduzir custos. O circuito é simples e de baixo custo. A utilização de um Arduino junto aos módulos permite maior eficiência e controle do sistema

em comparação a outros dispositivos do mercado que possuem código fechado e possuem um custo maior. Foi possível com a montagem do projeto obter informações precisas acerca da temperatura e da umidade dentro de uma sala rack comprovada por equipamentos de medição com uma faixa de diferença entre 1C° – 0.5C° na temperatura e 2% na medição da umidade.

Um controle eficaz e de baixo custo tem uma grande valorização na indústria, proporcionando segurança e confiabilidade na proteção dos equipamentos. Ao utilizar um Arduino com módulos que podem ser trocadas trará muitos benefícios com destaque em simplicidade.

REFERÊNCIAS

ARDUINO, **What is Arduino?**, 2018.
Disponível em: https://www.arduino.cc/en/Guide/Introduction. Acesso em: 20 abr. 2021.

BLOGELETROGATE**, Sensor DHT11/AM2303**, 2020.
Disponível em: https://blog.eletrogate.com/sensores-dht11-DHT11/. Acesso em: 19 abr. 2021.

CISCO, **O que é um data center,** 2021.
Disponível em: https://www.cisco.com/c/pt_br/solutions/data-center-virtualization/what-is-a-data-center.html. Acesso em: 20 jun. 2021.

DRAGINO, **Whats is Dragino project?,** 2021.
Disponível em: https://www.dragino.com/support/faqs.html. Acesso em: 07 ago. 2021.

IBM, **Critérios de Design Ambiental**, 2021.
Disponível em: https://www.ibm.com/docs/pt-br/power7?topic=planning-environmental-design-criteria. Acesso em: 26 nov. 2021.

LAMBERTS, Roberto. et al., **Towards a Brazilian Standard on Thermal Comfort.** Florianópolis: UFSC - Engenharia Civil, 2013.

LOGPYX, **Módulo LoRa**, 2021.
Disponível em: https://logpyx.com/lora-solucao-para-conectividade-das-coisas/. Acesso em: 21 abr. 2021.

LUIS, F. D. M, **Plataforma de baixo custo para monitoramento de clima e automação do meio agropecuário.** Lajeado: UNIVATES - Engenharia da Computação, 2020.

LORA DEVELOPERS, **What are LoRa® and LoRaWAN®?**
Disponível em: https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/. Acesso em: 22 jun. 2021.

MELLO, D. A. P. **Solução para Monitoramento Ambiente Utilizando Arduíno.** Guarapuava: UTFPR - Tecnologia em Sistemas para Internet, 2016.

RIBEIRO, J. M. T. **Uma aplicação da tecnologia LoRa em um ambiente hospitalar**. Florianópolis: UFSC – Engenharia Eletrônica, 2019.

SCHWAB, A. L. **Criação de uma rede LoRa para projetos de pesquisa e desenvolvimento.** Novo Hamburgo: FEEVALE - Ciência da Computação, 2020.

TEIXEIRA, G. B.; ALMEIDA, J. V. P. **Rede LoRa® e protocolo LoRaWAN® aplicados na agricultura de precisão no Brasil.** Ponta Grossa: UTFPR - Engenharia Eletrônica, 2017.

THINGSPEAK, **ThingSpeak**, 2021.
Disponível em: https://www.mathworks.com/help/thingspeak/. Acesso em: 04 nov. 2021.

TINKERCAD, **About**, 2021.
Disponível em https://blog.tinkercad.com/. Acesso em: 20 jun. 2021.