**SÃO PAULO TECH SCHOOL**

TechSolutions

Manual de Instalação (DHT11)

Ana Catarina | RA: 01232106

Emmily Ferreira | RA: 01232079

Gustavo Ferreira | RA: 01232014

Kauã Vidal | RA: 01232026

Kauan Parente | RA: 01232154

Michelly Mendes | RA: 01232188

Renan Alves | RA: 01232172

São Paulo

2023

**Introdução**

“DHT11” é um sensor de umidade que captura dados de temperatura e umidade que a realiza leituras de temperaturas entre 0 a 50 Celsius e umidade entre 20% a 90%, nessa documentação, temos como prioridade detalhar suas funcionalidades e em caso de falha ou ‘extravio’ de nosso produto, o cliente possa entrar em contato realizando a abertura de um chamado e assim efetuar as mudanças necessárias.

**Montagem do Sensor DHT11**

São necessários alguns materiais e equipamento para montagem e configurações básicas para que o mesmo possa ler, e mostrar os dados corretamente. Para a utilização do sensor “DHT11”, devemos ter uma placa chamada “Arduino UNO” que é uma Microcontrolador baseado no ATmega328  ([datasheet](http://adrobotica.com/wp-content/uploads/2018/01/datasheet-atmega-328p.pdf)).Ele tem 14 pinos de entrada/saída digital (dos quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um cristal oscilador de 16MHz, uma conexão USB, uma entrada de alimentação uma conexão ICSP e um botão de reset.

Arduino Uno R3 Atmega 328p-pu



CARACTERISTICAS:

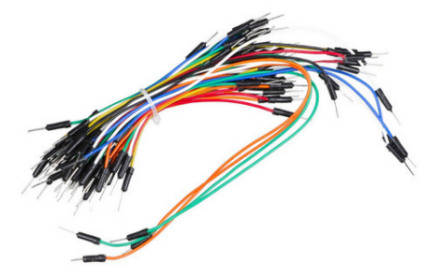
- Cabo USB para Arduino;   
- Essencial para programação do Arduino;   
- Compatível com Arduino Uno, Uno SMD, Mega e ADK.

ESPECIFICAÇÕES:

- Cor: Azul;   
- Extensão do cabo: ~30cm; Cabo USB tipo macho  
- Peso: 26g.

A Protoboard, também conhecida como placa de ensaio, matriz de contato, é uma placa que permite de modo simples e pratico a montagens de circuitos eletrônicos sem a necessidade da efetuação de soldas, precisando apenas “espetar” os componentes na placa.

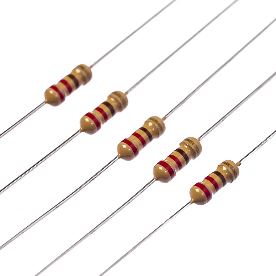
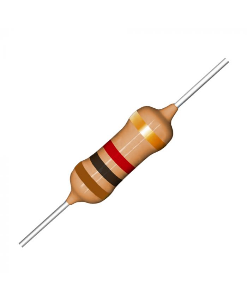
Mini Protoboard



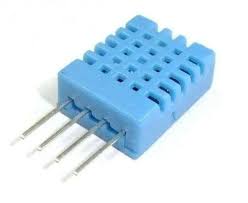
Os Cabos Jumper são fios de ligação elétrica, ideais e indispensável para quem faz montagens com Placa Arduino, Protoboard, Shields Arduino e os mais diversos Módulos, assim como para outras plataformas de desenvolvimento.

 Kit Jumper 65peças

O resistor é um componente com a função básica de limitar a passagem de corrente de um circuito. Resistores não possuem polaridade, logo podem ser usados em corrente alternada ou contínua. Em circuitos e esquemas elétricos.

     
Resistor 220r Resistor 10k Resistor1k 

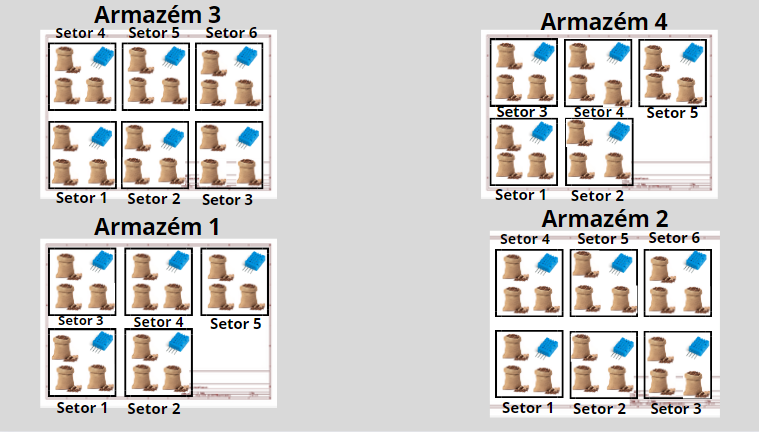
Em relação ao sensor utilizado o DHT11, temperatura pode variar de 10 a 90 % relativo à umidade e 0 à 50°C de temperatura. O sensor tem uma precisão em sua escala linear de +/- 5% da umidade, +/- 2ºC. Esta variação não necessariamente precisa ser implementada em software. Vamos considerar inicialmente uma variação de temperatura ambiente, entre 18ºC a 25ºC e a umidade 20% a 80%. Dependendo da aplicação do projeto, esta faixa deverá ser ajustada.



 DHT11 Modulo Sensor de Temperatura e Umidade

**Infraestrutura**

Para a instalação dos sensores é necessário o agendamento prévio do cliente através de nosso site pelo fale conosco onde será feito todos o processo de agendamento para visita técnica ao local do armazém sendo analisado os locais estratégicos para instalação e equipamento já presente no local que viabilize a obtenção de dados, bem como conexões elétricas para o bom funcionamento do equipamento. Sendo assim será solicitado a planta baixa do local, assim como o modelo abaixo apresentado para efetivar seus pontos de instalação.

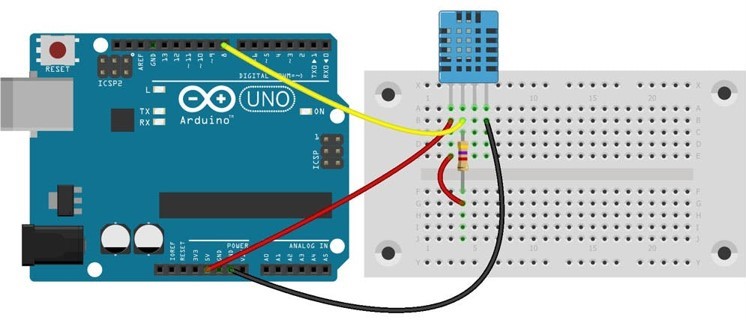


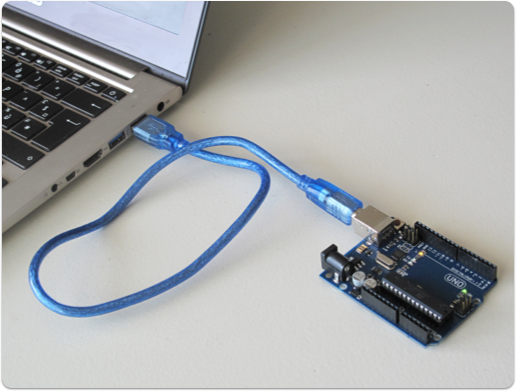
**Procedimento de Instalação dos sensores**

* Identifique os setores do armazém, garantindo que cada setor tenha uma distância de aproximadamente 10 metros entre eles.
* Certifique-se de que há uma fonte de alimentação próxima para conectar os sensores.
* Conecte cada sensor DHT11 a uma placa de prototipagem utilizando fios de conexão.
* Certifique-se de identificar claramente cada sensor, associando-o ao setor correspondente.
* Escolha uma posição adequada na parede de cada setor para fixar o sensor, preferencialmente na vertical. Utilize parafusos e buchas para garantir uma fixação segura.
* Antes de prosseguir, certifique-se de que todas as conexões estejam firmes.
* Realize uma verificação rápida para garantir que os sensores estejam recebendo energia.
* Após a instalação, monitore os dados fornecidos pelos sensores para garantir que estejam operando corretamente.
* Verifique se a leitura de umidade está de acordo com as condições esperadas.
* Realize verificações regulares para garantir que os sensores estejam funcionando adequadamente.
* Substitua qualquer sensor defeituoso imediatamente.

Observações:

* Certifique-se de seguir as normas de segurança ao lidar com eletricidade.
* Mantenha registros detalhados de instalação, manutenção e substituições para referência futura.
* Conexão com o sensor.  A seguir:





Conexão via USB

**Banco de Dados**

A configuração do banco de dados é adaptada de acordo com as necessidades específicas de armazenamento de café, visando oferecer um controle preciso dos dados adquiridos, permitindo identificar a localização de cada tipo de café nos armazéns e monitorar as condições ideais de umidade para garantir a qualidade e frescor do café armazenado.

Dessa forma, o banco de dados é estruturado para registrar de maneira eficiente, além de, possibilitar o acompanhamento em tempo real, das condições ambientais, assegurando que a umidade esteja sempre alinhada com os padrões ideais para a preservação das características únicas de cada variedade de café. Essa abordagem contribui significativamente para a gestão eficaz do estoque de café, otimizando os processos de monitoramento e garantindo a qualidade do produto final. No modelo abaixo podemos visualizar a maneira que seria representado no banco de dados os registros dos sensores, de acordo com o seu identificador, modelo, armazém correspondente, data e registro, além da umidade registrada.

Tabela

Descrição gerada automaticamente

**API**

* Faça o download dos arquivos necessários para o Arduino e o sensor.
* Abra o terminal na pasta do arquivo do Arduino utilizando Git Bash Here ou qualquer outra ferramenta que você esteja usando.
* No terminal, digite o comando npm install e aguarde a conclusão da instalação dos pacotes necessários definidos pelo Node.js.
* Prossiga com a atualização de bibliotecas e faça as configurações necessárias no script conforme as instruções fornecidas.
* Após a conclusão da atualização e configuração, abra novamente o terminal na pasta do projeto e insira o comando npm start para iniciar o projeto.

**Configurando o script do arduino**

O código abaixo serve para obter os dados do sensor DTH11, onde são conectados os fios em cada portal ou barramento e conectado via USB no laptop.

O mesmo deve ser adicionado na tela de ARDUINO IDE, já pré-instalado em seu computador.

#include <DHT.h>

#define DHTPIN A1

DHT dht(DHTPIN, DHT11);

void setup() {

pinMode(DHTPIN, INPUT);

Serial.begin(9600);

dht.begin();

}

void loop() {

/\* Definifição para o Sensor de Temperatura e Umidade \*/

float dht11\_umidade = dht.readHumidity();

float dht11\_temperatura = dht.readTemperature();

Serial.print(dht11\_umidade);

Serial.print(";");

Serial.print(dht11\_temperatura);

Serial.println();

delay(1000);

}

|  |
| --- |
| * #include <DHT.h>: Esta linha inclui a biblioteca do sensor DHT. Isso permite que você se comunique com sensores da série DHT, como DHT11, DHT22, etc. * #define DHTPIN A1: Esta linha define o pino ao qual o pino de dados do sensor DHT está conectado. Neste caso, está definido como A1. * DHT dht(DHTPIN, DHT11);: Esta linha declara um objeto DHT chamado dht e o inicializa com o pino de dados (DHTPIN) e o tipo de sensor (DHT11). Você pode mudar DHT11 para DHT22 se estiver usando um sensor DHT22. * void setup() {...}: A função setup é chamada uma vez quando o Arduino é iniciado. Ela inicializa o sensor DHT e configura a comunicação serial. * pinMode(DHTPIN, INPUT);: Configura o pino do sensor DHT como entrada. * Serial.begin(9600);: Inicializa a comunicação serial com uma taxa de transmissão de 9600 bps. * dht.begin();: Inicializa o sensor DHT. * void loop() {...}: A função loop é executada continuamente após a função setup. Ela lê os valores de umidade e temperatura do sensor DHT e os imprime no monitor serial. * float dht11\_umidade = dht.readHumidity();: Lê o valor de umidade do sensor DHT e o armazena na variável dht11\_umidade. * float dht11\_temperatura = dht.readTemperature();: Lê o valor de temperatura do sensor DHT e o armazena na variável dht11\_temperatura. * Serial.print(dht11\_umidade);: Imprime o valor de umidade no monitor serial. * Serial.print(";");: Imprime um ponto e vírgula como separador. * Serial.print(dht11\_temperatura);: Imprime o valor de temperatura no monitor serial. * Serial.println();: Imprime um caractere de nova linha para passar para a próxima linha no monitor serial. * delay(1000);: Pausa o programa por 1000 milissegundos (1 segundo) antes da próxima iteração. |

**Configuração do main.js**

O main.js utiliza a biblioteca serialport, para ler dados de um Arduino conectado a uma porta USB e, em seguida, insere os dados em nosso banco MySQL. Além disso, ele cria um servidor Express para fornecer um ponto de extremidade da API para recuperar os dados do sensor. Abaixo será representado o código que realizará essas funções, com o passo a passo:

const serialport = require('serialport');

const express = require('express');

const mysql = require('mysql2');

const SERIAL\_BAUD\_RATE = 9600;

const SERVIDOR\_PORTA = 3000;

const HABILITAR\_OPERACAO\_INSERIR = true; /\*Sempre verdadeiro\*/

const serial = async (

    valoresDht11Umidade

) => {

    const poolBancoDados = mysql.createPool(

        {

            host: '', /\*local do SQL\*/

            port: 3306,  /\* a porta de entrada do usb, entre no cmd do pc para ativar, "netstat -ano", pegue o PID ex:"12567" e faça "taskkill /PID 12567 -F" obs: veja em qual porta está seu arduino, aparece no git bash quando dá npm start.\*/

            user: '', /\* usuario do SQL\*/

            password: '', /\*senha do usuario\*/

            database: '' /\* database do banco, INSERT INTO sensores (dht11\_umidade, dht11\_temperatura, luminosidade, lm35\_temperatura, chave),\*/

        }

    ).promise();

    const portas = await serialport.SerialPort.list();

    const portaArduino = portas.find((porta) => porta.vendorId == 2341 && porta.productId == 43);

    if (!portaArduino) {

        throw new Error('O arduino não foi encontrado em nenhuma porta serial');

    }

    const arduino = new serialport.SerialPort(

        {

            path: portaArduino.path,

            baudRate: SERIAL\_BAUD\_RATE

        }

    );

    arduino.on('open', () => {

        console.log(`A leitura do arduino foi iniciada na porta ${portaArduino.path} utilizando Baud Rate de ${SERIAL\_BAUD\_RATE}`);

    });

    arduino.pipe(new serialport.ReadlineParser({ delimiter: '\r\n' })).on('data', async (data) => {

        const valores = data.split(';');

        const dht11Umidade = parseFloat(valores[0]);

        valoresDht11Umidade.push(dht11Umidade);

        console.log(dht11Umidade)

        if (HABILITAR\_OPERACAO\_INSERIR) {

            await poolBancoDados.execute(

                'INSERT INTO registro (dado, fkSensor) VALUES (?, ?)',

                [dht11Umidade, 1]

            );

            var dht11Umidade2 = dht11Umidade + 5;

            console.log(dht11Umidade2)

            await poolBancoDados.execute(

                'INSERT INTO registro (dado, fkSensor) VALUES (?, ?)',

                [dht11Umidade2, 2]

            );

        }

    });

    arduino.on('error', (mensagem) => {

        console.error(`Erro no arduino (Mensagem: ${mensagem}`)

    });

}

const servidor = (

    valoresDht11Umidade

) => {

    const app = express();

    app.use((request, response, next) => {

        response.header('Access-Control-Allow-Origin', '\*');

        response.header('Access-Control-Allow-Headers', 'Origin, Content-Type, Accept');

        next();

    });

    app.listen(SERVIDOR\_PORTA, () => {

        console.log(`A API executada com sucesso na porta ${SERVIDOR\_PORTA}`);

    });

    app.get('/sensores/dht11/umidade', (\_, response) => {

        return response.json(valoresDht11Umidade);

    });

}

(async () => {

    const valoresDht11Umidade = [];

    await serial(

        valoresDht11Umidade

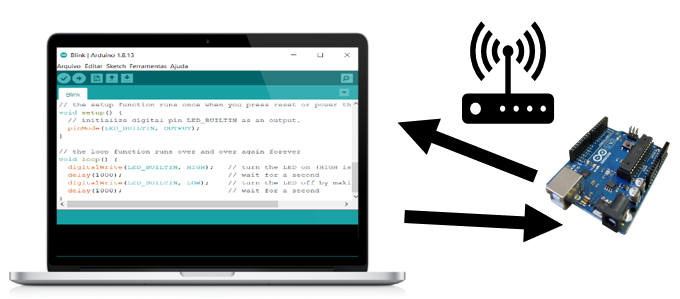
    );

    servidor(

        valoresDht11Umidade

    );

})();



Conexão como código IDE

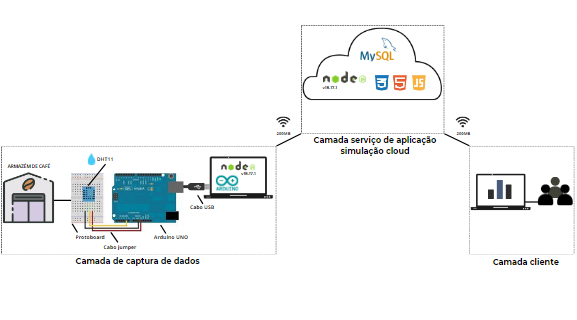


Diagrama de Solução Técnico.