

Sensor de Temperatura Digital com Termistor KY-001

O módulo KY-001 é composto por um sensor digital de temperatura DS18B20 para Arduino, Raspberry Pi, e ESP32. Este sensor é caracterizado por fornecer leituras de temperatura de 9 a 12 bits, realiza medições na faixa de -55° a 125°C, em ambiente seco, úmido ou submerso, não necessitando de um componente externo para isso, além de apresentar os valores em graus celsius.

O DS18B20 conta com precisão de $\pm 0,5^{\circ}\text{C}$ na faixa de medição de -10°C a 85°C , outrossim, apresenta faixas de medição distintas, podendo ter o incremento de $0,5^{\circ}\text{C}$ (9 bits), $0,25^{\circ}\text{C}$ (10 bits), $0,125^{\circ}\text{C}$ (11 bits) e $0,0625^{\circ}\text{C}$ (12 bits), sendo essa última a resolução padrão. O sensor de temperatura vem equipado com uma função de alarme, com valores acionadores programáveis.

As especificações resumidas consistem em:

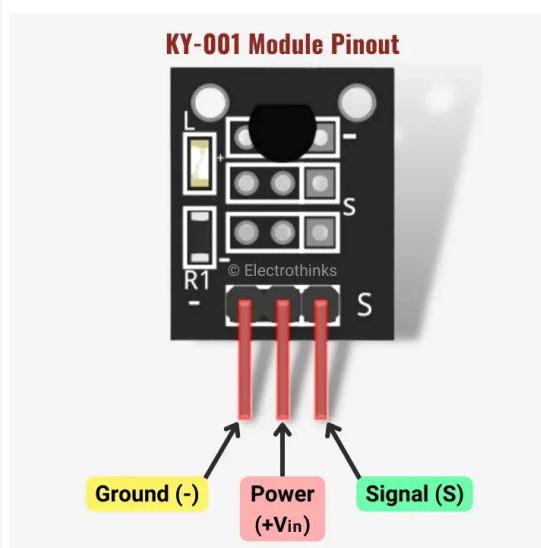
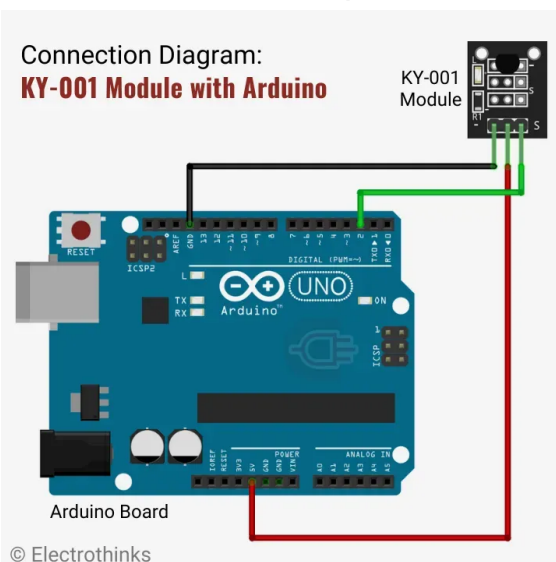
- **Chipset:** DS18B20
- **Voltagem de Operação:** 3.0V até 5.5V
- **Comunicação:** One-Wire Bus
- **Alcance de Medição de Temperatura:** -55°C to 125°C
- **Acurácia de Medição de Temperatura:** $\pm 0.5\text{C}$ (entre -10°C até 85°C)
- **Dimensão da Placa:** 18.5mm x 15mm

Este sensor pode ser usado para calcular a temperatura de soluções líquidas ou ambientes rígidos, em outras palavras, como um termômetro, ou para ativar dispositivos sensíveis ao calor, como o termovelocímetro, os quais são comuns em armazéns de combustível.

CONEXÃO:

Os componentes do projeto são:

- 1 Arduino Uno Rev 3
- 3 jumpers para as conexões
- 1 Sensor de Temperatura KY-001



CÓDIGO:

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup(void)
{
  Serial.begin(9600);
  sensors.begin();
}

void loop(void)
{
  sensors.requestTemperatures();
  Serial.println(sensors.getTempCByIndex(0));
  delay(2000);
}
```

OUTPUT(Exemplo em um ambiente de 24°C):

```
24
24
24
24
24
24
```

O código necessita das bibliotecas *One Wire* e *Dallas Temperature*. A biblioteca *One Wire* serve para prover comunicação entre o sensor de temperatura e a biblioteca *Dallas Temperature*, a qual proporciona a temperatura sendo registrada no sensor, em Celsius ou Fahrenheit.

CARREGAR O OUTPUT EM UM ARQUIVO

Logo abaixo segue um código, o qual armazena 10 vezes o *output* da *serial*, em um arquivo .csv, e o momento em que o dado foi guardado. Nós necessitamos de instalar o módulo *pyserial*, e importar as bibliotecas *datetime*, *time*, *serial*(do módulo *pyserial*) para a realização do código abaixo.

```

#Importação das bibliotecas necessárias
import serial
import csv
from datetime import datetime
import time

#Conexão com a Serial

serial_port = 'COM1'
baud_rate = 9600
serial_timeout = 2
ser = serial.Serial(serial_port, baud_rate, timeout = serial_timeout)

#Criação de um arquivo .csv, e armazenamento de 10 dados
CSV_FILE = 'temperatura.csv'
with open(CSV_FILE, 'w', newline='') as file:

    writer = csv.writer(file)
    writer.writerow(['id', 'temperatura', 'ano', 'mes', 'dia', 'hora',
'minuto', 'segundo'])
    i = 0

    while i <= 10:
        id = i
        data = ser.readline().decode().strip()
        ano = datetime.now().year
        mes = datetime.now().month
        dia = datetime.now().day
        hora = datetime.now().hour
        minuto = datetime.now().minute
        segundo = datetime.now().second

        writer.writerow([id, data, ano, mes, dia, hora, minuto, segundo])
        i = i+1
        time.sleep((1000000-datetime.now().microsecond)/1000000)
ser.close()
file.close()

```

ARQUIVO .CSV GERADO (EXEMPLO):

id	temperatura	ano	mes	dia	hora	minuto	segundo
0	24,2023	6	29	0	45	40	
1	24,2023	6	29	0	45	41	
2	24,2023	6	29	0	45	42	
3	24,2023	6	29	0	45	43	
4	24,2023	6	29	0	45	44	
5	24,2023	6	29	0	45	45	
6	24,2023	6	29	0	45	46	
7	24,2023	6	29	0	45	47	
8	24,2023	6	29	0	45	48	
9	24,2023	6	29	0	45	49	
10	24,2023	6	29	0	45	50	

Carregar os dados em um Banco de Dados(PostgreSQL)

Em seguida, se encontra um código que armazena 10 vezes o *output* da *serial*, em uma tabela de um banco de dados *PostgreSQL*, e o momento em que o dado foi guardado. Nós necessitamos de instalar o módulo *pyserial*, para a leitura dos dados da *serial*, e o módulo *psycopg2* para a conexão com o banco de dados. Neste código criamos uma tabela chamada *temperatura*, com a coluna *id* como chave primária, e mais 7 colunas, que consistem em 'temperatura'(output da *serial*), ano, mês, dia, hora, minuto e segundo.

```
#Importação das bibliotecas necessárias
import serial
import psycopg2
from datetime import datetime

#Conexão com a Serial
serial_port = 'COM1'
baud_rate = 9600
serial_timeout = 2
ser = serial.Serial(serial_port, baud_rate, timeout = serial_timeout)

#Conexão com o banco de dados
conn = psycopg2.connect(
    host="dataiesb.iesbtech.com.br",
    port=5432,
    dbname="postgres",
    user="2212120009_Matheus_Porfirio",
    password="2212120009"
)

csr = conn.cursor()

#Criação da tabela na base de dados
csr.execute("""
    CREATE TABLE temperatura (
        id SERIAL PRIMARY KEY,
        temperatura VARCHAR(3),
        year VARCHAR(4),
        month VARCHAR(2),
        day VARCHAR(2),
        hour VARCHAR(2),
        minute VARCHAR(2),
        second VARCHAR(2)
    )
""")
conn.commit()

data2 = []
j = 0
while j <= 10:
```

```

id = str(j)
temp = str(ser.readline().decode().strip())
ano = str(datetime.now().year)
mes = str(datetime.now().month)
dia = str(datetime.now().day)
hora = str(datetime.now().hour)
minuto = str(datetime.now().minute)
segundo = str(datetime.now().second)

data = (id, temp, ano, mes, dia, hora, minuto, segundo)
data2.append(data)
j = j+1

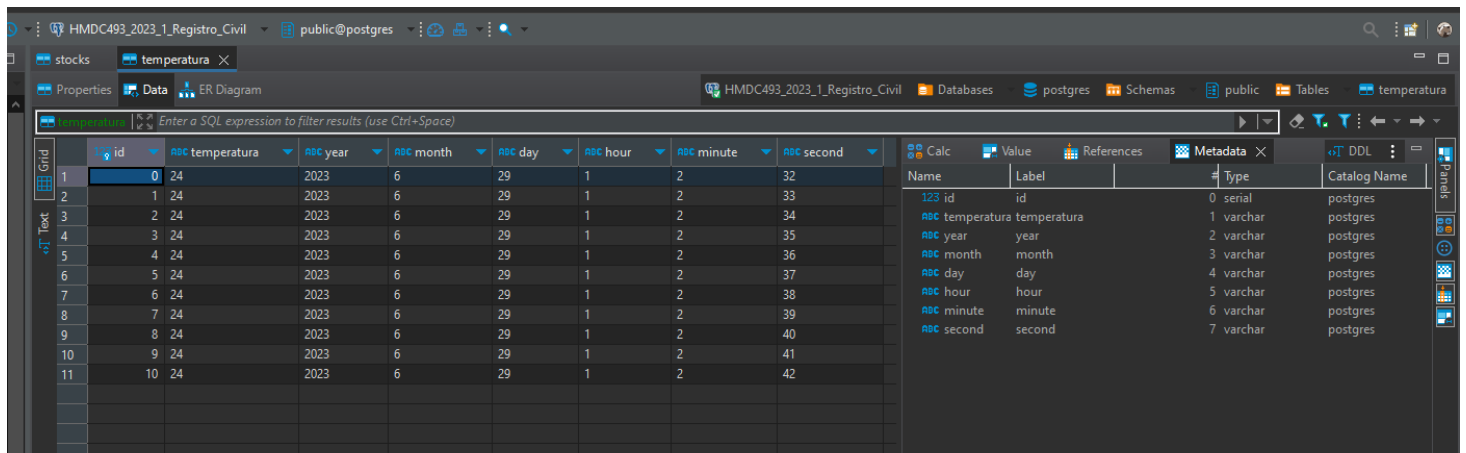
args = ','.join(ser.mogrify("(%s,%s,%s,%s,%s,%s,%s,%s)",
i).decode('utf-8'))

for i in data2)

csr.execute("INSERT INTO temperatura VALUES " + (args))
conn.commit()
time.sleep((1000000-datetime.now().microsecond)/1000000)
data2 = []

```

Tabela no Banco de Dados(Exemplo):



id	temperatura	year	month	day	hour	minute	second
0	24	2023	6	29	1	2	32
1	24	2023	6	29	1	2	33
2	24	2023	6	29	1	2	34
3	24	2023	6	29	1	2	35
4	24	2023	6	29	1	2	36
5	24	2023	6	29	1	2	37
6	24	2023	6	29	1	2	38
7	24	2023	6	29	1	2	39
8	24	2023	6	29	1	2	40
9	24	2023	6	29	1	2	41
10	24	2023	6	29	1	2	42

Dessa forma, podemos ter um armazenamento de temperatura preciso, e constante, o que poderia ser empregado para armazenar dados de um experimento, o qual necessita de uma vigilância de longo prazo de algum elemento ou de um ambiente, e a variação de sua temperatura.