TUTORIAL

O sensor DHT11 é um sensor de temperatura e umidade, que permite medir temperaturas na faixa de 0 a 50 Celsius, e umidade entre de 20 a 90%.

Sua faixa de precisão para temperatura é de aproximadamente 2 graus, e de umidade, por volta de 5%.

O sensor em si tem 4 pinos, mas o pino 3 não é utilizado:

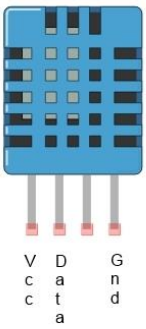
O mais comum é encontrá-lo em forma de módulo, onde temos apenas 3 pinos: Vcc, Data e Gnd:

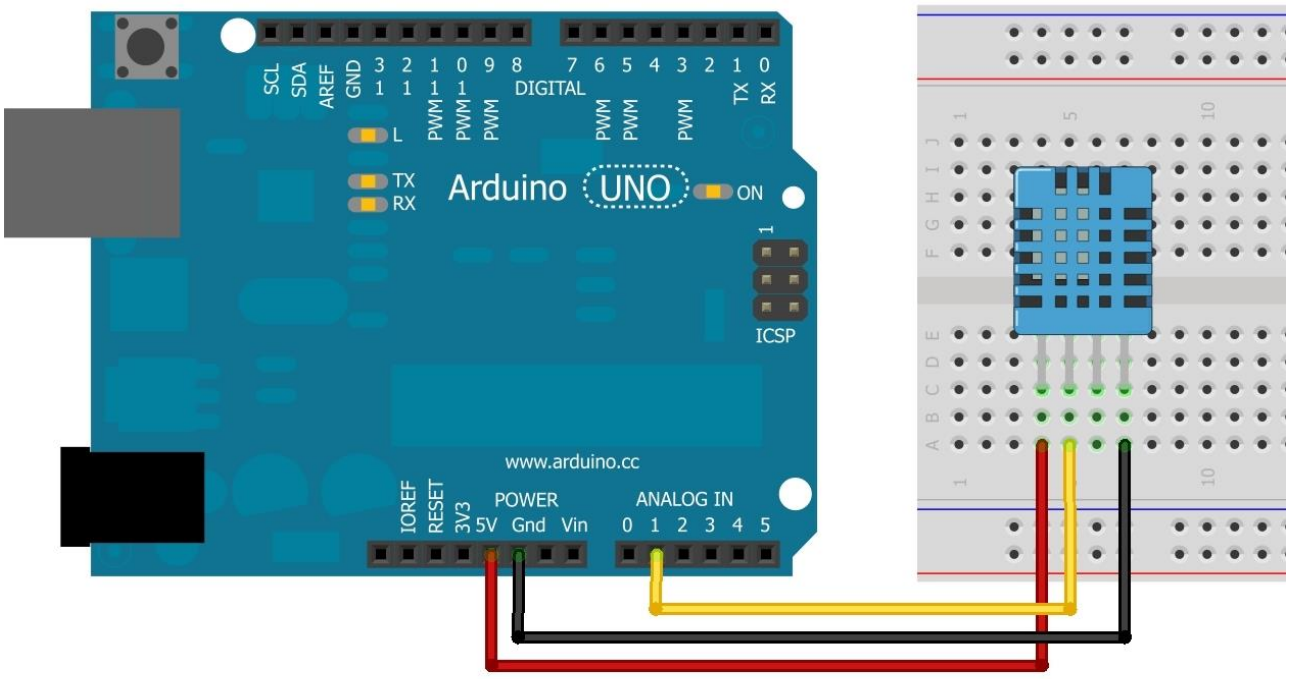
Para montarmos o circuito seguiremos os seguintes passos:

1. Utilizando (01) um Arduino, (01) uma protoboard, (01) um sensor de temperatura, (03) três jumpers e (01) um microprocessador Arduino,

montemos o circuito seguindo o esquemático abaixo e a numeração a seguir na placa do Arduino:

- *Conecte pino 1 do sensor (esquerda) ao +5V;*
- *Conecte pino 2 do sensor ao pino de dados definido em seu Arduino;*
- *Conecte pino 4 do sensor ao GND*
- *Conecte o resistor de 10K entre pin 2 (dados) e ao pino 1 (VCC) do sensor;*





2. Utilizando o ambiente de desenvolvimento do Arduino, copiamos o código abaixo (presente na pasta **setup_DHT11**) e solicitamos para verificar:

```
#include "Arduino.h"
#define DEBUG_PRINTER Serial
#ifdef DHT_DEBUG
#define DEBUG_PRINT(...) { DEBUG_PRINTER.print(__VA_ARGS__); }
#define DEBUG_PRINTLN(...) { DEBUG_PRINTER.println(__VA_ARGS__); }
#else
#define DEBUG_PRINT(...) {}
#define DEBUG_PRINTLN(...) {}
#endif
```

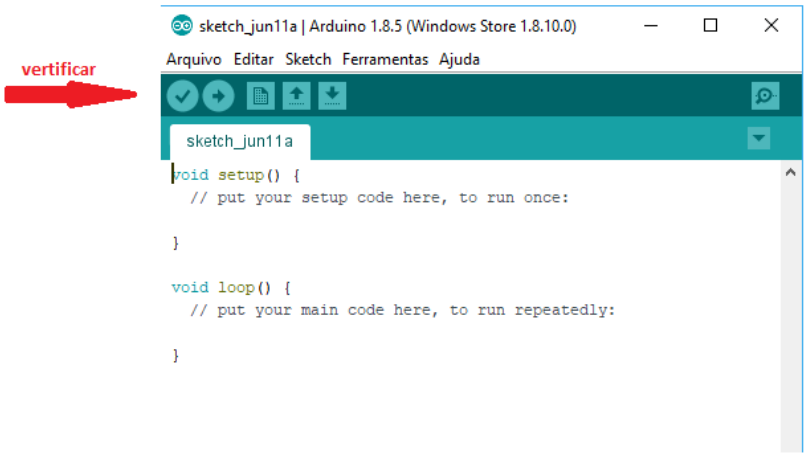
```
#define DHT11 11 //modelo de sensor
#define DHT22 22 //modelo de sensor
#define DHT21 21 //modelo de sensor
#define AM2301 21 //modelo de sensor
#define DHTPIN 2 // saida de dados digital do arduino nr 2
#define DHTTYPE DHT11 // define o modelo DHT 11, caso use outro sensor deve trocar o por DHT 22/DHT 21/AM2301


class DHT {
  public:
    DHT(uint8_t pin, uint8_t type, uint8_t count = 6);
    void begin(void);
    float Ler_Temperatura(bool S = false, bool force = false);
    float convertC_F(float);
    float convertF_C(float);
    float computeHeatIndex(float temperatura, float percentHumidade, bool ehFahrenheit = true);
    float Ler_humidade(bool force = false);
    boolean read(bool force = false);

  private:
    uint8_t data[5];
    uint8_t _pin, _type;
#ifdef __AVR
    uint8_t _bit, _port;
#endif
    uint32_t _ult_time, _maxcycles;
    bool _ult_result;
    uint32_t expectPulse(bool level);
};

class InterruptLock {
  public:
    InterruptLock() {
      noInterrupts();
    }
    ~InterruptLock() {
      interrupts();
    }

};

#define MIN_INTERVAL 2000

DHT::DHT(uint8_t pin, uint8_t type, uint8_t count) {
  _pin = pin;
  _type = type;
#ifdef __AVR
  _bit = digitalPinToBitMask(pin);
  _port = digitalPinToPort(pin);
#endif
  _maxcycles = microsecondsToClockCycles(1000);
}

void DHT::begin(void) {
  pinMode(_pin, INPUT_PULLUP);
  _ult_time = -MIN_INTERVAL;
  DEBUG_PRINT("Max clock cycles: "); DEBUG_PRINTLN(_maxcycles, DEC);
}

//boolean S == Scale.  True == Fahrenheit; False == Celcius
float DHT::Ler_Temperatura(bool S, bool force) {
  float f = NAN;

  if (read(force)) {
    switch (_type) {
      case DHT11:
```

```
        f = data[2];
        if (S) {
          f = convertC_F(f);
        }
        break;
      case DHT22:
      case DHT21:
        f = data[2] & 0x7F;
        f *= 256;
        f += data[3];
        f *= 0.1;
        if (data[2] & 0x80) {
          f *= -1;
        }
        if (S) {
          f = convertC_F(f);
        }
        break;
    }
  }
  return f;
}

float DHT::convertC_F(float c){
  return c * 1.8 + 32;
}

float DHT::convertF_C(float f) {
  return (f - 32) * 0.55555;
}

float DHT::Ler_humidade(bool force) {
  float f = NAN;
  if (read()) {
    switch (_type) {
      case DHT11:
        f = data[0];
        break;
      case DHT22:
      case DHT21:
        f = data[0];
        f *= 256;
        f += data[1];
        f *= 0.1;
        break;
    }
  }
  return f;
}

float DHT::computeHeatIndex(float temperatura, float percentHumidade, bool ehFahrenheit) {

  float hi;

  if (!ehFahrenheit)
    temperatura = convertC_F(temperatura);

  hi = 0.5 * (temperatura + 61.0 + ((temperatura - 68.0) * 1.2) + (percentHumidade * 0.094));

  if (hi > 79) {
    hi = -42.379 +
         2.04901523 * temperatura +
         10.14333127 * percentHumidade +
         -0.22475541 * temperatura * percentHumidade +
         -0.00683783 * pow(temperatura, 2) +
         -0.05481717 * pow(percentHumidade, 2) +
         0.00122874 * pow(temperatura, 2) * percentHumidade +
```

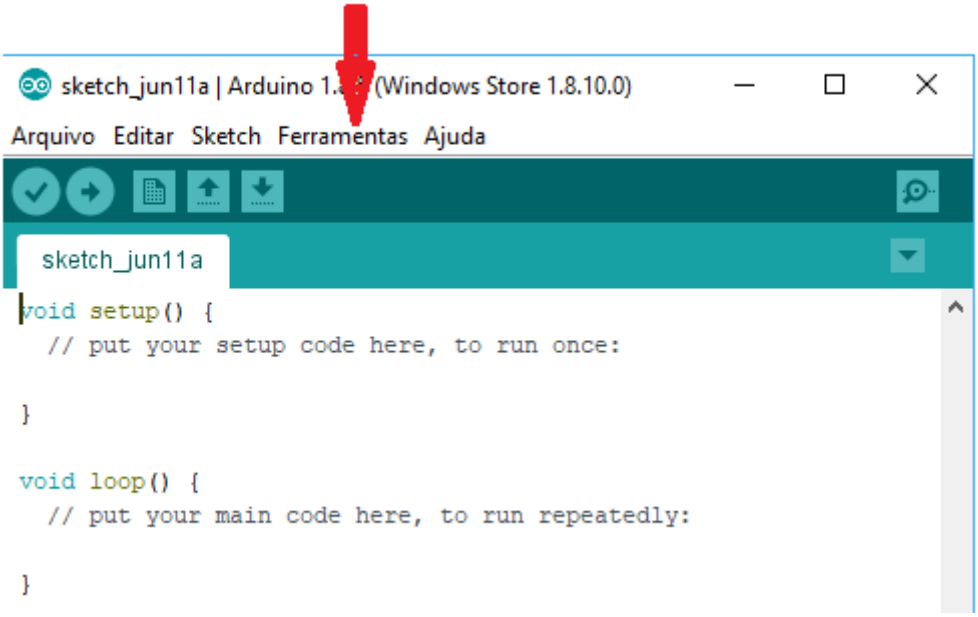
```
        0.00085282 * temperatura * pow(percentHumidade, 2) +
        -0.00000199 * pow(temperatura, 2) * pow(percentHumidade, 2);

    if ((percentHumidade < 13) && (temperatura >= 80.0) && (temperatura <= 112.0))
      hi -= ((13.0 - percentHumidade) * 0.25) * sqrt((17.0 - abs(temperatura - 95.0)) * 0.05882);

    else if ((percentHumidade > 85.0) && (temperatura >= 80.0) && (temperatura <= 87.0))
      hi += ((percentHumidade - 85.0) * 0.1) * ((87.0 - temperatura) * 0.2);
  }

  return ehFahrenheit ? hi : convertF_C(hi);
}

boolean DHT::read(bool force) {

  uint32_t currenttime = millis();
  if (!force && ((currenttime - _ult_time) < 2000)) {
    return _ult_result; // return last correct measurement
  }
  _ult_time = currenttime;


  data[0] = data[1] = data[2] = data[3] = data[4] = 0;

  digitalWrite(_pin, HIGH);
  delay(250);


  pinMode(_pin, OUTPUT);
  digitalWrite(_pin, LOW);
  delay(20);

  uint32_t cycles[80];
  {

    InterruptLock lock;

    digitalWrite(_pin, HIGH);
    delayMicroseconds(40);

    pinMode(_pin, INPUT_PULLUP);
    delayMicroseconds(10);  // Delay a bit to let sensor pull data line low.


    if (expectPulse(LOW) == 0) {
      DEBUG_PRINTLN(F("Timeout waiting for start signal low pulse."));
      _ult_result = false;
      return _ult_result;
    }
    if (expectPulse(HIGH) == 0) {
      DEBUG_PRINTLN(F("Timeout waiting for start signal high pulse."));
      _ult_result = false;
      return _ult_result;
    }


    for (int i = 0; i < 80; i += 2) {
      cycles[i]   = expectPulse(LOW);
      cycles[i + 1] = expectPulse(HIGH);
    }
  }
  for (int i = 0; i < 40; ++i) {
    uint32_t lowCycles  = cycles[2 * i];
    uint32_t highCycles = cycles[2 * i + 1];
    if ((lowCycles == 0) || (highCycles == 0)) {
      DEBUG_PRINTLN(F("Timeout waiting for pulse."));
      _ult_result = false;
```

```
      return _ult_result;
    }
    data[i / 8] <<= 1;

    if (highCycles > lowCycles) {

      data[i / 8] |= 1;
    }

  }

  DEBUG_PRINTLN(F("Received:"));
  DEBUG_PRINT(data[0], HEX); DEBUG_PRINT(F(", "));
  DEBUG_PRINT(data[1], HEX); DEBUG_PRINT(F(", "));
  DEBUG_PRINT(data[2], HEX); DEBUG_PRINT(F(", "));
  DEBUG_PRINT(data[3], HEX); DEBUG_PRINT(F(", "));
  DEBUG_PRINT(data[4], HEX); DEBUG_PRINT(F(" =? "));
  DEBUG_PRINTLN((data[0] + data[1] + data[2] + data[3]) & 0xFF, HEX);

  if (data[4] == ((data[0] + data[1] + data[2] + data[3]) & 0xFF)) {
    _ult_result = true;
    return _ult_result;
  }
  else {
    DEBUG_PRINTLN(F("Checksum failure!"));
    _ult_result = false;
    return _ult_result;
  }
}

uint32_t DHT::expectPulse(bool level) {
  uint32_t count = 0;

#ifdef __AVR
  uint8_t portState = level ? _bit : 0;
  while ((*portInputRegister(_port) & _bit) == portState) {
    if (count++ >= _maxcycles) {
      return 0; // Exceeded timeout, fail.
    }
  }

#else
  while (digitalRead(_pin) == level) {
    if (count++ >= _maxcycles) {
      return 0; // Exceeded timeout, fail.
    }
  }
#endif

  return count;
}

// Conecte pino 1 do sensor (esquerda) ao +5V
// Conecte pino 2 do sensor ao pino de dados definido em seu Arduino
// Conecte pino 4 do sensor ao GND
// Conecte o resistor de 10K entre pin 2 (dados)
// e ao pino 1 (VCC) do sensor
DHT dht(DHTPIN, DHTTYPE);
 void setup() {
  Serial.begin(9600);

  dht.begin();
}

void loop() {
  delay(500);
```

```
  int h = dht.Ler_humidade();
  int t = dht.Ler_Temperatura();
  int f = dht.Ler_Temperatura(true);

  char x = Serial.read();

  if (x == 'a') {
    Serial.print(h);
    Serial.print(",");
    Serial.print(t);
    Serial.print(",");
    Serial.println(f);
  }
}
```



3. Vá em **ferramentas** no item **placa** e escolha o modelo da sua placa e depois em **processador** e escolha o modelo caso exista mais de um, por último e como o Arduino já conectado ao PC, selecione a **porta COM**, logo abaixo do item **processador**.



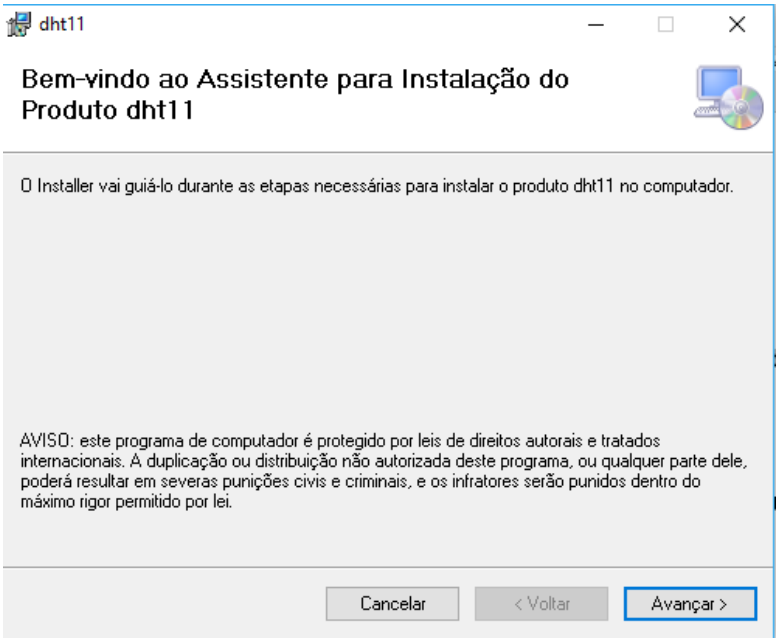
4. Por último, carregue o programa no Arduino e o sensor DHT11 enviará automaticamente dados para o PC.

TUTORIAL (INTERFACE GRÁFICA)

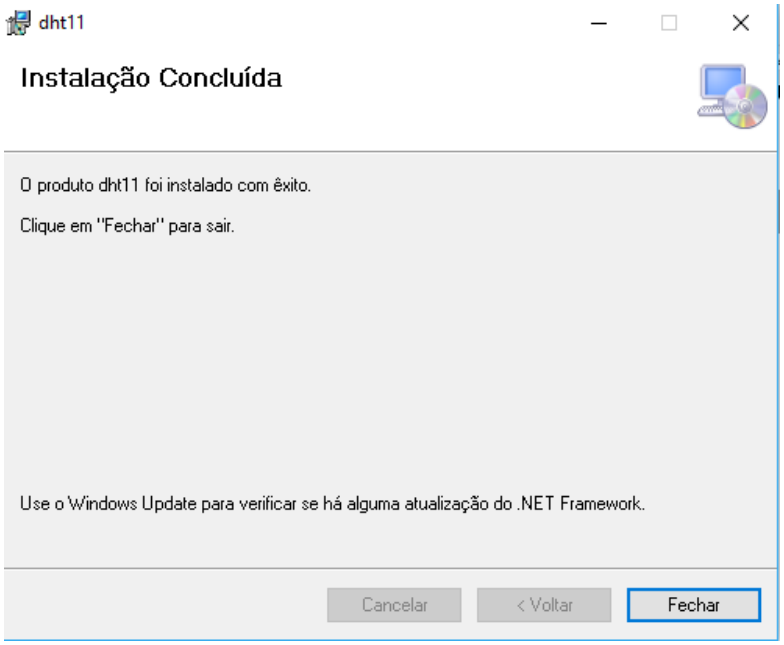
Com a instalação e conecção do DHT11 ao Arduino e sua transmissão de dados ao PC, partiremos para instalação da interface gráfica do sensor.

Para isso seguiremos o seguinte roteiro:

1. Na pasta **setup_DHT11,** acesse a pasta **dht11,** clique em **release** e depois em **setup** ou **dht11.**
2. Seguindo as telas, clicaremos em **avançar** até a última tela

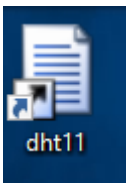


3. Abaixo a ultima tela, onde clicaremos em **fechar**

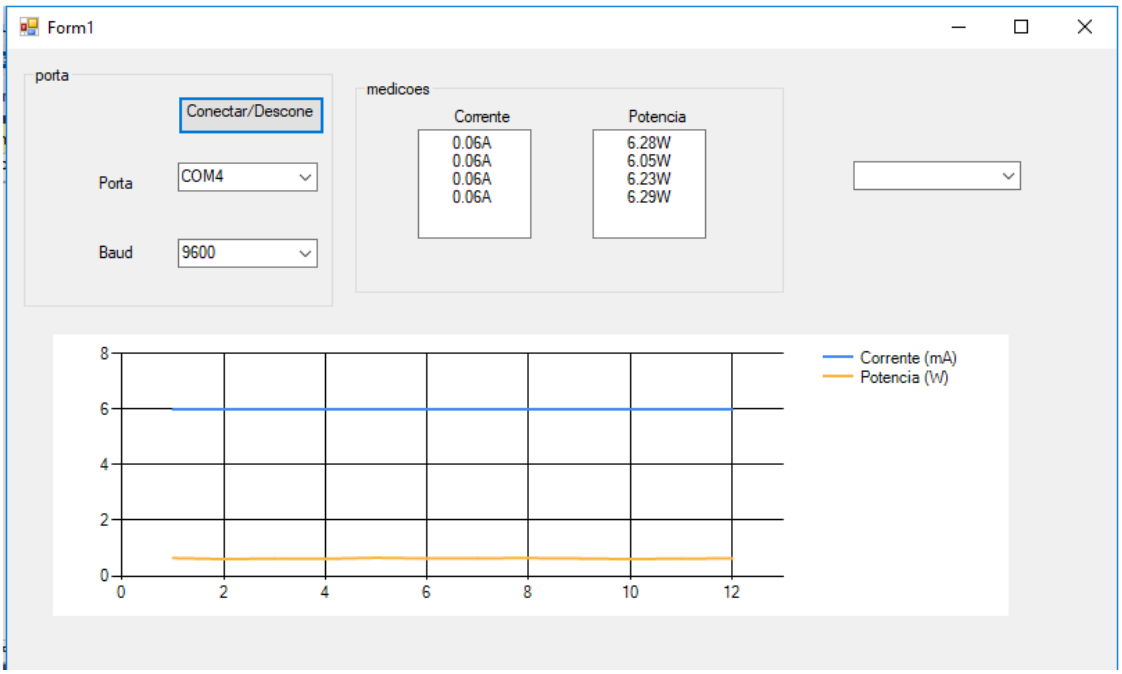


UTILIZAÇÃO

Com o Arduino e o sensor instalado e já funcionado, clicaremos no ícone semelhante à figura abaixo:



Partindo desse ícone abrirá uma janela semelhante a esta:



Primeiramente escolheremos a Porta Serial, no entanto o programa já está configurado para mostrar a porta serial que o Arduino escolher, então apenas aceite a opção que ele te oferecer.

Posteriormente, você deverá escolher a taxa de Bauds que para Arduino Uno e Mega é da ordem de 9600.

Por último, clique em **Conectar/Desconectar** e em seguida você perceberá que o gráfico e as medições de temperatura e umidade aparecerão automaticamente na tela.

Para sair, clique em **Conectar/Desconectar** e depois no **X** do canto superior direito.