|  |  |
| --- | --- |
|  | TutoRIEL  workshop mini-station météo avec arduino |

# Préambule

Les sources sont disponibles sur <https://github.com/jbcotard/FabLab_workshop_meteo_arduino>

Le maker a à sa disposition plusieurs types de capteurs de température / humidité, dont voici une liste (non exhaustive) :

|  |  |  |  |
| --- | --- | --- | --- |
| **LM35** | **TMP36** | **DHT11** | **DHT22** |
| Brochage d'un capteur LM35 | https://2.bp.blogspot.com/-RRYnMgxszXs/TfnIoAAZCGI/AAAAAAAABA0/mMC2Li3E_OU/s200/Tmp36.gif |  |  |
| * Capteur analogique de température * Types :   - LM35DZ : 0 à 100°C (+/- 1.5°C)  - LM35CZ : -40 à 110°C (+/- 1.5°C)  - LM35CAZ : -40 à 110°C (+/- 1°C)   * Environ 1.20 € (DZ) | * Capteur analogique de température * -50 à +150 °C * Environ 2.50€ | * Capteur analogique de température et d’humidité * 0 à +50 °C (+/- 2°C) * Humidité : 0 ~ 100 % * Environ 2€ | * Capteur analogique de température et d’humidité * -40 à +150 °C (+/- 0.5°C) * Humidité : 20 ~ 80% * Environ 4.50€ |

# Etape 1 : Montage du capteur sur une carte arduino

## Prérequis :

installer le studio arduino IDE (<https://www.arduino.cc/en/Main/Software>)

Remarque : il existe une version en ligne, avec des limitations (<https://create.arduino.cc/editor>)

## Matériels nécessaire :

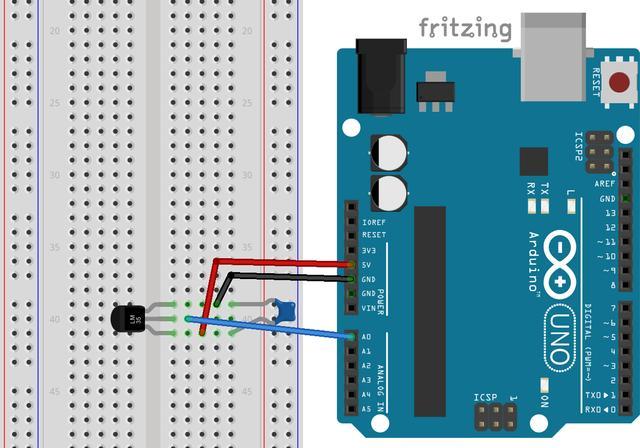
|  |  |  |
| --- | --- | --- |
| **LM35DZ** | **TMP36** | **DHT11** |
| * Une carte Arduino UNO (et son câble USB), * Un capteur LM35DZ * Un condensateur de 100nF (optionnel, mais recommandé), * Une plaque d'essai et des fils pour câbler notre montage. | * Une carte Arduino UNO (et son câble USB), * Un capteur TMP36 * Un condensateur de 100nF (optionnel, mais recommandé), * Une plaque d'essai et des fils pour câbler notre montage. | * Une carte Arduino UNO (et son câble USB), * Un capteur DHT11 * Une résistance de 4.7K ohms – code couleur jaune / violet / rouge, * Un condensateur de 100nF, * Une plaque d'essai et des fils pour câbler notre montage.   N.B. Si vous disposez d'un module précâblé, la résistance et le condensateur sont déjà être câblés |
|  |  |  |

## Montage :

### Montage pour LM35DZ

Source : <https://www.carnetdumaker.net/articles/mesurer-une-temperature-avec-un-capteur-lm35-et-une-carte-arduino-genuino/>

#### Le schéma de montage



#### Le code

<https://github.com/jbcotard/FabLab_workshop_meteo_arduino/tree/master/01-montage-arduino-sensor/LM35DZ>

*/\**

*\* Code d'exemple pour le capteur LM35 (2°C ~ +110°C).*

*\*/*

*// Fonction setup(), appelée au démarrage de la carte Arduino*

void setup() {

*// Initialise la communication avec le PC*

Serial.begin(9600);

}

*// Fonction loop(), appelée continuellement en boucle tant que la carte Arduino est alimentée*

void loop() {

*// Mesure la tension sur la broche A0*

int valeur\_brute = analogRead(A0);

*// Transforme la mesure (nombre entier) en température via un produit en croix*

float temperature\_celcius = valeur\_brute \* (5.0 / 1023.0 \* 100.0);

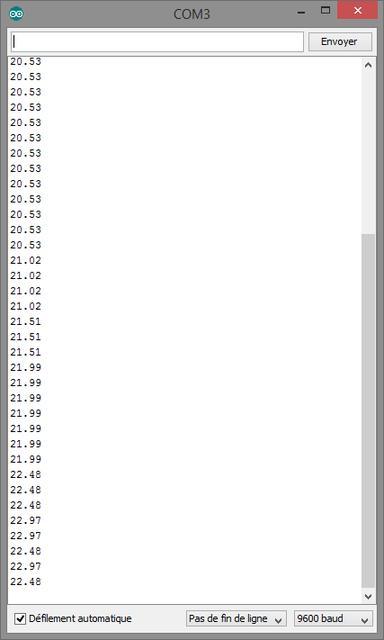
*// Envoi la mesure au PC pour affichage et attends 250ms*

Serial.println(temperature\_celcius);

delay(250);

}

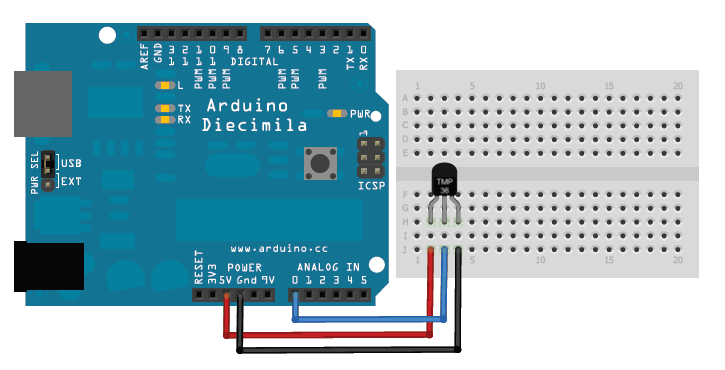
#### Le résultat dans le moniteur de trace



### Montage pour TMP36

Source : <https://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor>

#### Le schéma de montage



#### Code

<https://github.com/jbcotard/FabLab_workshop_meteo_arduino/tree/master/01-montage-arduino-sensor/TMP36>

//TMP36 Pin Variables

int sensorPin = 0; //the analog pin the TMP36's Vout (sense) pin is connected to

//the resolution is 10 mV / degree centigrade with a

//500 mV offset to allow for negative temperatures

/\*

\* setup() - this function runs once when you turn your Arduino on

\* We initialize the serial connection with the computer

\*/

void setup()

{

Serial.begin(9600); //Start the serial connection with the computer

//to view the result open the serial monitor

}

void loop() // run over and over again

{

//getting the voltage reading from the temperature sensor

int reading = analogRead(sensorPin);

// converting that reading to voltage, for 3.3v arduino use 3.3

float voltage = reading \* 5.0;

voltage /= 1024.0;

// print out the voltage

Serial.print(voltage); Serial.println(" volts");

// now print out the temperature

float temperatureC = (voltage - 0.5) \* 100 ; //converting from 10 mv per degree wit 500 mV offset

//to degrees ((voltage - 500mV) times 100)

Serial.print(temperatureC); Serial.println(" degrees C");

// now convert to Fahrenheit

float temperatureF = (temperatureC \* 9.0 / 5.0) + 32.0;

Serial.print(temperatureF); Serial.println(" degrees F");

delay(1000); //waiting a second

}

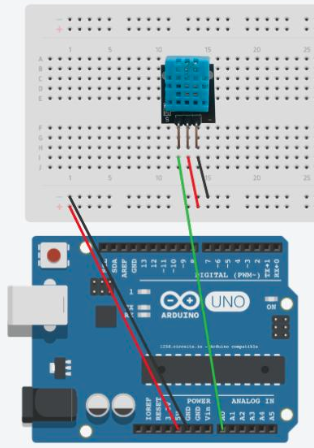
#### Le résultat dans le moniteur de trace

TBC

### Le montage pour DHT11

Source : <https://www.carnetdumaker.net/articles/utiliser-un-capteur-de-temperature-et-dhumidite-dht11-dht22-avec-une-carte-arduino-genuino/>

#### Schéma du montage



#### Code

<https://github.com/jbcotard/FabLab_workshop_meteo_arduino/tree/master/01-montage-arduino-sensor/DHT11>

*/\*\**

*\* Exemple de code pour la lecture d'un capteur DHT11 ou DHT22.*

*\*/*

*/\*\* Broche "DATA" du capteur \*/*

**const** byte BROCHE\_CAPTEUR = 5;

*/\* Code d'erreur de la fonction readDHT11() et readDHT22() \*/*

**const** byte DHT\_SUCCESS = 0; *// Pas d'erreur*

**const** byte DHT\_TIMEOUT\_ERROR = 1; *// Temps d'attente dépassé*

**const** byte DHT\_CHECKSUM\_ERROR = 2; *// Données reçues erronées*

*/\*\* Fonction setup() \*/*

void setup() {

*/\* Initialisation du port série \*/*

Serial.begin(115200);

Serial.println(F("Demo DHT11 et DHT22"));

*/\* Place la broche du capteur en entrée avec pull-up \*/*

pinMode(BROCHE\_CAPTEUR, INPUT\_PULLUP);

}

*/\*\* Fonction loop() \*/*

void loop() {

float temperature, humidity;

*/\* Lecture de la température et de l'humidité, avec gestion des erreurs \*/*

*// N.B. Remplacer readDHT11 par readDHT22 en fonction du capteur utilisé !*

**switch** (readDHT11(BROCHE\_CAPTEUR, &temperature, &humidity)) {

**case** DHT\_SUCCESS:

*/\* Affichage de la température et du taux d'humidité \*/*

Serial.print(F("Humidite (%): "));

Serial.println(humidity, 2);

Serial.print(F("Temperature (^C): "));

Serial.println(temperature, 2);

**break**;

**case** DHT\_TIMEOUT\_ERROR:

Serial.println(F("Pas de reponse !"));

**break**;

**case** DHT\_CHECKSUM\_ERROR:

Serial.println(F("Pb de communication !"));

**break**;

}

*/\* Pas plus d'une mesure par seconde \*/*

*// N.B. Avec le DHT22 il est possible de réaliser deux mesures par seconde*

delay(1000);

}

…

#### Résultat dans le moniteur de trace

TBC

# Etape 2 : Afficher la température sur un afficheur

## Prérequis :

il sera nécessaire d’ajouter des bibliothèques dans le studio Arduino IDE :

par exemple MD\_MAX72xx



## Matériels nécessaire :

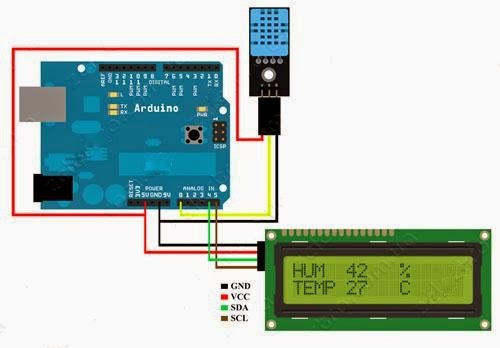
|  |  |
| --- | --- |
| **LCD I2C** | **MATRIX LED** |
| * Une carte Arduino UNO (et son câble USB), * Un capteur température * LCD i2c 16x2 * Une plaque d'essai et des fils pour câbler notre montage. | * Une carte Arduino UNO (et son câble USB), * Un capteur température * Bande 4 Matrix LED 8x8, * Une plaque d'essai et des fils pour câbler notre montage. |
| IIC I2C Série 1602 16x2 LCD Module de Caractères Display Bleu Pour Arduino Accs | MagiDeal MAX7219 LED Dot Matrix Module d'Affichage Numérique 8 X 8 pour Arduino |

## Montage :

### Le montage pour LCD

#### PREREQUIS

#### Schéma du montage



#### Code

#include <dht.h>

#include <Wire.h>

#include <LCD.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x20, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); // Addr, En, Rw, Rs, d4, d5, d6, d7, backlighpin, polarity

#define dht\_dpin A0 //no ; here. Set equal to channel sensor is on

dht DHT;

void setup(){

Serial.begin(9600);

delay(300);//Let system settle

//Serial.println("Humidity and temperature\n\n");

delay(700);//Wait rest of 1000ms recommended delay before

//accessing sensor

lcd.begin(16,2);

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print("Temp & Humidity");

}//end "setup()"

void loop(){

//This is the "heart" of the program.

DHT.read11(dht\_dpin);

lcd.setCursor(0, 1);

lcd.print(DHT.temperature);

lcd.setCursor(5, 1);

lcd.print("C");

lcd.setCursor(8, 1);

lcd.print(DHT.humidity);

lcd.setCursor(13, 1);

lcd.print("%");

//Serial.print("Current humidity = ");

//Serial.print(DHT.humidity);

//Serial.print("% ");

//Serial.print("temperature = ");

//Serial.print(DHT.temperature);

//Serial.println("C ");

delay(2000);//Don't try to access too frequently... in theory

//should be once per two seconds, fastest,

//but seems to work after 0.8 second.

}// end loop()

### Le montage pour MATRIX LED

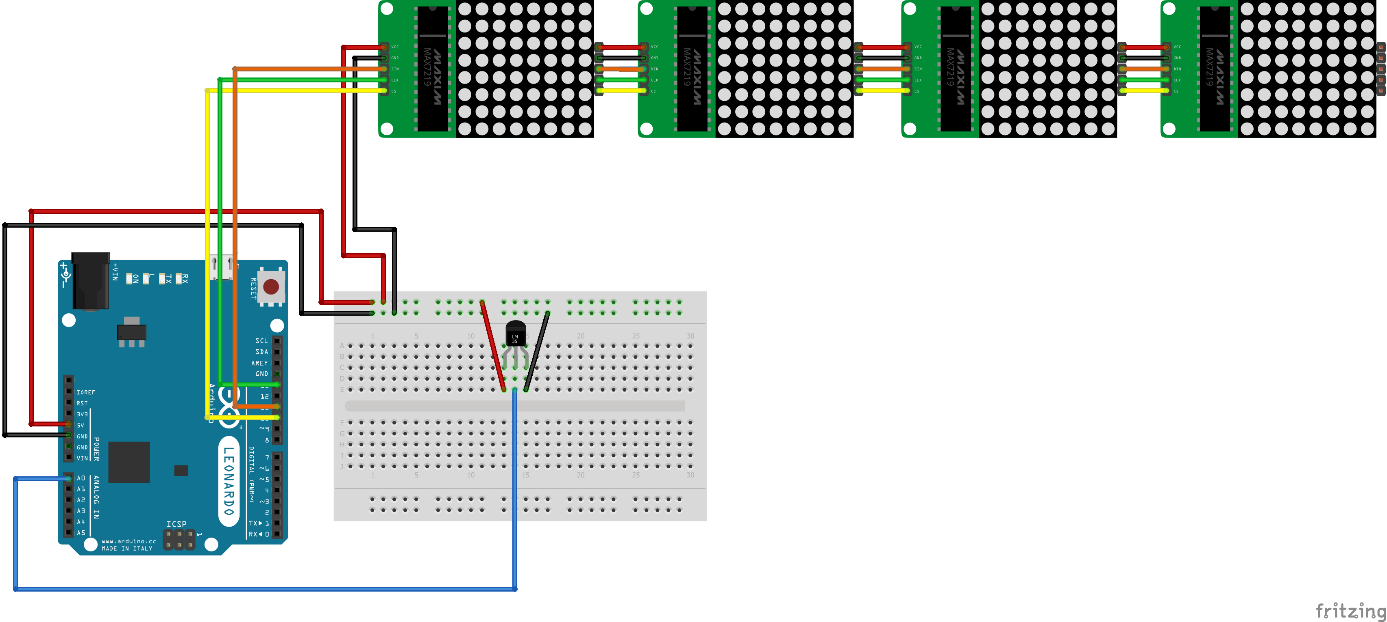
#### PREREQUIS

Il est nécessaire d’installer la bibliothèque « MD\_MAX72xx »

Note : en fonction du type de matrix LED, il faut modifier le parametre USE\_xxx\_HW du fichier .h.

#define USE\_FC16\_HW 1

#### Schéma du montage



#### Code

<https://github.com/jbcotard/FabLab_workshop_meteo_arduino/tree/master/02-montage-afficheur/LM35DZ_MATRIX>

#include <MD\_MAX72xx.h>

#define MAX\_DEVICES 4

#define CLK\_PIN 13

#define DATA\_PIN 11

#define CS\_PIN 10

MD\_MAX72XX mx = MD\_MAX72XX(DATA\_PIN, CLK\_PIN, CS\_PIN, MAX\_DEVICES);

// Text parameters

#define CHAR\_SPACING 1 // pixels between characters

int valeur\_brute;

int total = 0;

int nbmesures = 15; // nb secondes entre chaque affichage

void setup()

{

mx.begin();

//https://playground.arduino.cc/Main/LM35HigherResolution

analogReference(INTERNAL); // 2.5v sur leonardo

mx.control(MD\_MAX72XX::INTENSITY, 1);

// première lecture - un peu bidon

total = analogRead(A0) \* nbmesures;

}

void loop()

{

//float temperature\_celcius = valeur\_brute \* (5.5 / 1023.0 \* 100.0); // Sans analogReference(INTERNAL); /

float temperature\_celcius = (total/nbmesures)/4.096;

//float temperature\_celcius = (total / nbmesures) / 4.185; // T° Ludovic

String mystring = " " + String(temperature\_celcius) ;

mystring.remove(mystring.length() - 1);

char charBuf[50];

mystring.toCharArray(charBuf, 50);

printText(0, MAX\_DEVICES - 1, charBuf);

// Symbole "°"

mx.setPoint(0,3,true);

mx.setPoint(2,3,true);

mx.setPoint(1,4,true);

mx.setPoint(1,2,true);

total = 0;

for (int i=0;i<nbmesures; i++){

mx.setPoint(0, 0, true);

delay(100);

valeur\_brute = analogRead(A0);

total = total + valeur\_brute;

mx.setPoint(0, 0, false);

delay(900);

}

}

void printText(uint8\_t modStart, uint8\_t modEnd, char \*pMsg)

// Print the text string to the LED matrix modules specified.

// Message area is padded with blank columns after printing.

{

uint8\_t state = 0;

uint8\_t curLen;

uint16\_t showLen;

uint8\_t cBuf[8];

int16\_t col = ((modEnd + 1) \* COL\_SIZE) - 1;

mx.control(modStart, modEnd, MD\_MAX72XX::UPDATE, MD\_MAX72XX::OFF);

do // finite state machine to print the characters in the space available

{

switch (state)

{

case 0: // Load the next character from the font table

// if we reached end of message, reset the message pointer

if (\*pMsg == '\0')

{

showLen = col - (modEnd \* COL\_SIZE); // padding characters

state = 2;

break;

}

// retrieve the next character form the font file

showLen = mx.getChar(\*pMsg++, sizeof(cBuf) / sizeof(cBuf[0]), cBuf);

curLen = 0;

state++;

// !! deliberately fall through to next state to start displaying

case 1: // display the next part of the character

mx.setColumn(col--, cBuf[curLen++]);

// done with font character, now display the space between chars

if (curLen == showLen)

{

showLen = CHAR\_SPACING;

state = 2;

}

break;

case 2: // initialize state for displaying empty columns

curLen = 0;

state++;

// fall through

case 3: // display inter-character spacing or end of message padding (blank columns)

mx.setColumn(col--, 0);

curLen++;

if (curLen == showLen)

state = 0;

break;

default:

col = -1; // this definitely ends the do loop

}

} while (col >= (modStart \* COL\_SIZE));

mx.control(modStart, modEnd, MD\_MAX72XX::UPDATE, MD\_MAX72XX::ON);

}

# Etape 3 : Connecter à un Raspberry PI et Domoticz

TBC