Lycée TOUCHARD WASHINGTON

Programmation Orientée Objet

TD n°1

SNir 1
2021-2022

Thermomètre numérique

1- Installation de la librairie

Avec l'outil PlatformIO, vous allez ajouter la librairie permettant la gestion du capteur Dallas DS18S20 avec un ESP32.

Recherche de la librairie :

```
pio lib search DS18S20
```

OneWire
=======
#ID: 1
Control 1-Wire protocol (DS18S20, DS18B20, DS2408 and etc)

Keywords: onewire, 1-wire, bus, sensor, temperature, ibutton
Compatible frameworks: Arduino
Compatible platforms: Infineon XMC, Kendryte K210, GigaDevice GD32V, ASR Microelectronics ASR650x, Atmel AVR, Atmel SAM, Espressif 8266, Intel ARC32, Microchip PIC32, Nordic nRF51, ST STM32, Teensy, TI MSP430, TI TIVA, Espressif 32, Nordic nRF52, ST STM8, Atmel megaAVR, Logic Green boards, HWLogic, LOGICROM Development Platform, Raspberry Pi RP2040
Authors: Paul Stoffregen, Jim Studt, Tom Pollard, Derek Yerger, Josh Larios, Robin James, Glenn Trewitt, Jason Dangel, Guillermo Lovato, Ken Butcher, Mark Tillotson, Bertrik Sikken, Scott Roberts

DallasTemperature
================
#ID: **54**
Arduino Library for Dallas Temperature ICs (DS18B20, **DS18S20**, DS1822, DS1820)

Keywords: onewire, 1-wire, bus, sensor, temperature
Compatible frameworks: Arduino
Compatible platforms: Infineon XMC, Kendryte K210, GigaDevice GD32V, ASR Microelectronics ASR650x, Atmel AVR, Atmel SAM, Espressif 8266, Intel ARC32, Microchip PIC32, Nordic nRF51, ST STM32, Teensy, TI MSP430, TI TIVA, **Espressif 32**, Nordic nRF52, ST STM8, Atmel megaAVR, Logic Green boards, HWLogic, LOGICROM Development Platform, Raspberry Pi RP2040
Authors: Miles Burton, Tim Newsome, Guil Barros, Rob Tillaart

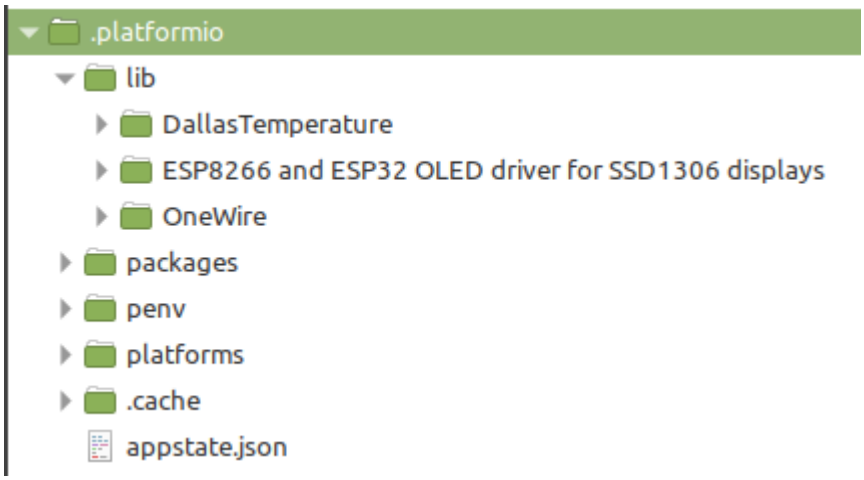
De nombreuses autres librairies permettant le pilotage capteur de température OneWire DS18S20 du fabricant Dallas. Le choix se porte sur la deuxième.

```
pio lib -g install 54
```

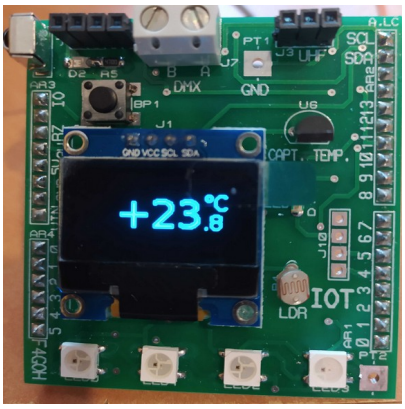
Elle s'installe dans votre dossier : **$${HOMEDIR}/.platformio/lib**

Voici le lien github pour l'obtenir par ailleurs. On y trouve également des exemples et de la documentation : https://github.com/milesburton/Arduino-Temperature-Control-Library

Votre répertoire .platformio/lib contient maintenant les éléments suivants, la librairie OneWire s'installe automatiquement.



2- Connexion de l'afficheur sur votre cible ESP32



Dans un premier temps, positionnez le cavalier J10-7 J11-7 pour relier le capteur de température de la carte ESP32 au microcontrôleur, broche GPIO18.

Par la suite, si votre capteur ne fonctionne pas, reliez la broche 12 de la carte Afficheur IOT à la broche J10-7 de la carte Equilibreuse ESP32 avec un câble femelle-femelle. Cette broche correspond à la broche GPIO18 de vote ESP32

3- Préparation du projet

Préparez un projet nommé **Thermometre** avec **PlatformIO** et votre environnement de développement.

```
pio project init --ide netbeans --board lolin32
```

Ouvrez le projet avec **NetBeans**. Pensez à modifier la commande **run** dans les propriétés du projet.

```
pio run --target upload
```

Enfin, ajoutez les deux nouvelles librairies dans les options du compilateur C++. Pour rappel, elles se trouvent directement dans le répertoire :

```
$${HOMEDIR}/.platformio/lib/OneWire
$${HOMEDIR}/.platformio/lib/DallasTemperature
```

4- Exemple d'utilisation de la librairie

Voici un premier exemple qui permet d'utiliser le capteur de température **DS18S20** sur le bus **OneWire** branché sur la **broche 18** de l'ESP32. Le constructeur de la classe **DallasTemperature** a besoin de connaître l'adresse de l'instance **oneWire** pour piloter le capteur.

```
#include <Arduino.h>
#include <OneWire.h>
#include <DallasTemperature.h> // pio lib -g install 54

OneWire oneWire(18); // broche GPIO18 de l'ESP32 pour OneWire
DallasTemperature capteur(&oneWire);

void setup()
{
    capteur.begin();
    Serial.begin(115200);
}

void loop()
{
    capteur.requestTemperatures();
    float tempC = capteur.getTempCByIndex(0);
    if(tempC != DEVICE_DISCONNECTED_C)
    {
        Serial.printf("Temperature %.1f\n\r",tempC);
    }
    else
    {
        Serial.println("La température ne peut pas être obtenue");
    }
    delay(5000);
}
```

Après initialisation du capteur dans la fonction **setup()**. Toutes les 5 secondes, la mesure de température est demandée au capteur avec la méthode **requestTemperatures()**. Comme il est possible de disposer de plusieurs capteurs de température sur le même bus, il est nécessaire de demander au capteur la température en degré Celsius à l'index 0 (index du 1er capteur, nous n'en avons qu'un seul sur notre bus). La méthode **getTempCByIndex()** réalise cette opération, si le résultat est différent de **DEVICE_DISCONNECTED_C** nous pouvons afficher la valeur.

Si ce programme ne fonctionne pas, votre capteur est vraisemblablement défectueux, utilisez celui de la carte Afficheur IOT.

5- Application

Ajoutez une nouvelle classe nommée **Afficheur** à votre projet. Cette classe doit hériter de la classe **SSD130006Wire** comme vu dans le cours.

```
#ifndef AFFICHEUR_H
#define AFFICHEUR_H
#include <Arduino.h>
#include <SSD1306Wire.h>
#include "font.h"
class Afficheur : public SSD1306Wire
{
public:
    Afficheur();
    void Initialiser();
    void AfficherTemperature(const int _temperature, const int _dixieme,
                             const uint16_t _dx=25, const uint16_t _dy=10);
};
#endif // AFFICHEUR_H
```

Le constructeur passe les paramètres à la classe **SSD130006Wire** suite à la relation d'héritage.

La fonction **Initialiser()** initialise l'afficheur et éventuellement retourner l'affichage verticalement.

La fonction **AfficherTemperature()** reprend le code réalisé lors du TD1 après modification pour tenir compte de l'affichage.

Modifiez la boucle principale pour afficher périodiquement sur l'afficheur la température lue par le capteur.

Pour séparer la partie entière et la partie décimale de la température, la librairie **math.h** propose la fonction suivante permettant d'arrondir vers le bas à la décimale la plus proche la valeur passée en paramètres.

```
double floor (double x);
```

Par un jeu de transtypage en entier, il est facile d'obtenir la partie entière. La partie décimale est obtenue en soustrayant le nombre de départ à la valeur entière et en multipliant par 10 le résultat.

Après avoir instancié de manière globale votre afficheur et l'avoir initialisé dans la fonction **setup(),** faites l'appel de la méthode **AfficherTemperature()** dans la fonction **loop()**.