UE INF-IoT
« Internet of Things »

Sujet de TP : Lecture de température via le standard *1-Wire*

L'objectif de cette séance de travaux pratiques est d'étudier comment un micro-contrôleur (MCU) ou micro-processeur (MPU) peut lire des capteurs lui fournissant des informations via le standard de bus *1-Wire*.

Dans le cas présent, vous allez utiliser un MCU (le modèle exact vous sera indiqué en cours de séance), et allez y connecter plusieurs capteurs via le standard *1-Wire*. Vous allez ensuite développer un petit programme permettant de lire ces capteurs.

1 Aperçu des capteurs DS18S20

Il existe quelques dizaines de modèles de capteurs et actionneurs fonctionnant sur bus 1-Wire [1]. Parmi tous ces modèles, les capteurs de température de type DS18S20, DS18B20, et DS1822 (voir figure 1) figurent parmi les plus communément utilisés. Les capteurs DS18S20 et DS18B20 permettent de mesurer la température ambiante avec une précision de ±0.5°C dans l'intervalle [-10°C..+85°C], et avec une précision moindre dans un intervalle plus large. Les capteurs DS18S20 offrent une résolution fixe de 9 bits, et les capteurs DS18B20 une résolution ajustable entre 9 et 12 bits. Les capteurs DS1822 offrent également une résolution ajustable, mais avec une précision moindre (±2°C).

Pour ce TP vous allez utiliser des capteurs de type DS18S20 [2].

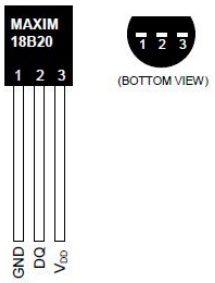

FIGURE 1 – Représentation d'un capteur DS18B20 (en conditionnement TO-92).

Chaque capteur *1-Wire* dispose d'un identifiant unique, exprimé sur 64 bits. Cet identifiant est fixé en usine et ne peut être modifié. C'est pourquoi les capteurs *1-Wire* peuvent aussi être utilisés comme jetons d'authentification.

Pour consulter un capteur précis sur un bus *1-Wire* (qui peut en porter un grand nombre) il faut pouvoir le désigner par son identifiant. Une procédure de découverte permet au maître d'un bus *1-Wire* de recenser tous les capteurs situés sur ce bus, afin de pouvoir s'adresser ensuite spécifiquement à l'un ou l'autre d'entre eux.

Chaque capteur *1-Wire* dispose d'une broche de transmission (broche DQ dans la figure 1) pouvant fonctionner en émission comme en réception. L'appellation *1-Wire* vient du fait qu'un seul fil (connecté à la broche DQ) est dédié aux transmissions. Les broches V_{DD} et GND servent à assurer l'alimentation électrique du composant, entre 3 V et 5.5 V.

Certains modèles de capteurs *1-Wire* peuvent fonctionner avec un mode d'alimentation dit « parasite ». Dans ce mode particulier les broches V_{DD} et GND sont couplées et reliées au circuit de masse (GND) du maître. Le capteur n'est alors relié au maître que par les circuits DQ et GND, et lorsque le maître place le circuit DQ à l'état haut les capteurs en profitent pour stocker une part de l'énergie ainsi reçue, ce qui leur donne suffisamment d'énergie pour fournir une réponse au maître.

1. https://github.com/owfs/owfs-doc/wiki/1Wire-Device-List
2. `https://datasheets.maximintegrated.com/en/ds/DS18S20.pdf`

2 Montage

Vous allez réaliser un montage simple pour connecter quelques (au moins deux) capteurs DS18S20 à un micro-contrôleur. Dans ce montage (voir figure 2) les capteurs seront alimentés en continu en 3.3 V. Ils ne fonctionneront donc pas en mode « parasite ».

Une résistance de 4.7 k■ sera insérée entre le circuit de transmission de données (broche DQ des capteurs DS18S20) et le circuit 3.3 V. Cette résistance, dite de *pullup*, assure que l'état au repos du bus *1-Wire* est l'état haut.

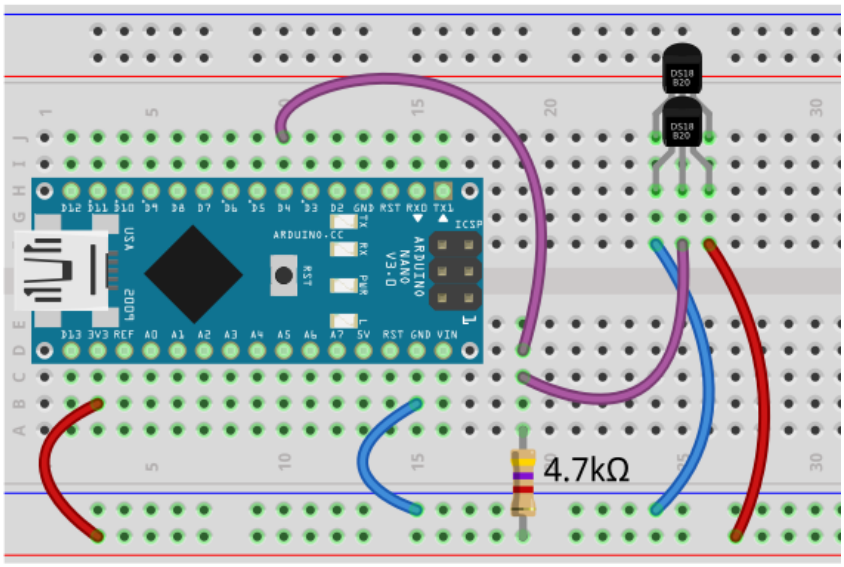

FIGURE 2 – Schéma du montage sur plaque d'expérimentation *(breadboard)*

3 Développement d'un programme de test

3.1 Objectif

En utilisant l'éditeur Visual Studio Code avec l'extension PlatformIO, vous allez développer un petit programme permettant :

1. d'identifier les capteurs connectés sur le bus lors de l'initialisation du micro-contrôleur, en ne retenant que les capteurs de type DS18S20 ;
2. de consulter périodiquement (par exemple toutes les 5 ou 10 secondes) l'état de chacun de ces capteurs, et d'afficher cet état via la console série du micro-contrôleur.

L'étape 1) définie ci-dessus devrait en toute logique être déclenchée dans la fonction `setup()` de votre programme, et l'étape 2) devrait l'être dans la fonction `loop()` (en veillant toutefois à respecter la périodicité choisie).

Dans l'étape 1) on ne cherchera à détecter que des capteurs de la famille DS18S20. Il existe d'autres types de capteurs capables de fonctionner sur un bus *1-Wire*, et de tels capteurs pourraient être déployés sur notre bus *1-Wire*. Par contre dans ce programme seuls les capteurs de température DS18S20 nous intéressent.

Ce programme sera développé en utilisant le Framework Arduino, et notamment la librairie `OneWire` [3] développée par Paul Stoffregen. Cette librairie fournit les fonctions élémentaires de communication *1-Wire*, mais il reste encore un peu de travail à faire pour consulter spécifiquement la température délivrée par un capteur DS18S20.

L'identification et la consultation des capteurs DS18S20 devront s'effectuer **uniquement** grâce aux fonctions de la librairie `OneWire` : vous n'êtes pas censé(e) utiliser une autre librairie spécialisée dédiée à la lecture de capteurs DS18S20 : de telles librairies existent, mais l'objectif est ici d'apprendre à vous en passer.

3. `https://github.com/PaulStoffregen/OneWire`

3.2 Démarche

Pour vous faire gagner du temps, le squelette du programme que vous devez réaliser est mis à votre disposition. Dans ce squelette les fonctions `setup()` et `loop()` sont déjà définies, ainsi que quelques fonctions annexes dont vous aurez besoin.

La figure 3 illustre le comportement attendu au niveau de la console.

Les commandes pouvant être adressées aux capteurs, ainsi que les réponses possibles, sont décrites dans la fiche technique [4] *(datasheet)* du capteur DS18S20.

Notez que par défaut le capteur DS18S20 fournit une température avec une résolution de 9 bits, soit 0.5°C, mais qu'il est possible d'affiner cette mesure en utilisant une méthode de calcul décrite dans la fiche technique. C'est pourquoi la figure 3 illustre des températures affichées à 0.1°C près.

```
Searching for sensors (DS18S20 only) on the 1-Wire bus
  2 sensors found

#0: 10 1E 4D B1 01 08 00 75
#1: 10 6D C2 D7 01 08 00 89

Starting measurement on all known sensors
Getting temperatures from all known sensors
#0: 20.3°C
#1: 20.2°C
```

FIGURE 3 – Illustration de la sortie du programme

3.3 Quelques remarques

— Dans la fonction `scan_bus()` vous veillerez à vérifier la somme de contrôle (CRC) des identifiants des capteurs détectés, afin d'ignorer tout identifiant invalide. Vous veillerez de même à ne retenir que les identifiants de capteurs de type DS18S20.

— Dans la fonction `start_measurement()` l'objectif est de déclencher la mesure de température simultanément sur tous les capteurs présents sur le bus, avec une seule commande. Il ne s'agit pas d'adresser successivement la même commande à chacun des capteurs. Cette autre option serait possible, mais cela prendrait plus de temps.

— Dans la fonction `get_temperature()` vous veillerez cette fois encore à vérifier la somme de contrôle (CRC) des données reçues du capteur. Si ces données sont invalides, alors la fonction devrait retourner `false`. De même si la température mesurée est de 85.0 °C : cette valeur est caractéristique d'une mesure erronnée de la part du capteur.

4. `https://datasheets.maximintegrated.com/en/ds/DS18S20.pdf`

Squelette du programme

Les fonctions `scan_bus()`, `start_measurement()`, et `get_temperature()` doivent être complétées.

```
1  #include <Arduino.h>
2  #include <OneWire.h>
3
4  #define ONE_WIRE_PIN    18
5  #define MAX_DEVICES     32
6
7  // Creation of a OneWire object to access the 1-Wire bus,
8  // using the specified digital pin
9  OneWire ds(ONE_WIRE_PIN);
10
11 // Sensor identifiers will be stored in 'ids', each identifier occupying
12 // 8 consecutive bytes in this array (so the id of sensor #0 is stored
13 // in ids[0..7], the id of sensor #1 in ids[8..15], etc.)
14 uint8_t ids[8 * MAX_DEVICES];
15
16 // Number of ids contained in 'ids' (cannot be greater than MAX_DEVICES)
17 int nb_ids = 0;
18
19
20 // ------------------------------------------------------------------------
21 // Display an array of bytes in hexadecimal representation
22 void show_bytes(uint8_t *addr, int nb_bytes) {
23
24   for (int i=0; i < nb_bytes; i++) {
25     if (addr[i] < 0x10) Serial.write('0');
26     Serial.print(addr[i], HEX);
27     Serial.print(' ');
28   }
29 }
30
31 // ------------------------------------------------------------------------
32 // Check the CRC in an array of bytes (assuming the last byte contains the CRC
33 // value for the rest of the array)
34 bool check_CRC(uint8_t *addr, int nb_bytes) {
35
36   return (OneWire::crc8(addr, nb_bytes - 1) == *(addr + nb_bytes - 1));
37 }
38
39 // ------------------------------------------------------------------------
40 // Display the ids of all known sensors
41 void show_all_ids() {
42
43   for (int i=0; i < nb_ids; i++) {
44     int offset=i * 8;
45     Serial.print('#'); Serial.print(i); Serial.print(": ");
46     show_bytes(ids + offset, 8);
47     Serial.println();
48   }
49 }
50
51 // ------------------------------------------------------------------------
52 // Scan the 1-Wire bus and identify all DS18S20 sensors on this bus
53 int scan_bus() {
54
```

```
55    Serial.println("TO BE COMPLETED");
56
57  }
58
59  // ------------------------------------------------------------------------
60  // Start measurement on all known sensors by issuing the same command to all
61  // sensors simultaneously
62  bool start_measurement() {
63
64    Serial.println("TO BE COMPLETED");
65  }
66
67  // ------------------------------------------------------------------------
68  // Get the temperature from sensor #i, and set the value of 'temp' accordingly
69  // Return 'true' if the temperature was obtained, and 'false' otherwise
70  bool get_temperature(int i, float *temp) {
71
72    Serial.println("TO BE COMPLETED");
73  }
74
75  // ------------------------------------------------------------------------
76  void setup() {
77
78    Serial.begin(9600); // Initalize the serial console
79
80    while (! Serial) {}; // Wait for the serial console to open
81
82    // Scan the 1-Wire until at least one DS18S20 sensor has been found
83    scan_bus();
84    while (nb_ids == 0) {
85      delay(5000);
86      scan_bus();
87    }
88
89    // Display the identifiers of all DS18S20 sensors
90    show_all_ids();
91    Serial.println();
92  }
93
94  // ------------------------------------------------------------------------
95  void loop() {
96
97    // Send a 'Start convert' command to all DS18S20 sensors so they start
98    // measuring the temperature
99    Serial.println(F("Starting measurement on all known sensors"));
100   start_measurement();
101
102   delay(800);  // Wait while temperature is being measured
103
104   // For each known sensor, get the temperature that has just been measured,
105   // and display this temperature
106   Serial.println(F("Getting temperatures from all known sensors"));
107   float temp;
108   for (int i=0; i < nb_ids; i++) {
109     Serial.print('#'); Serial.print(i); Serial.print(": ");
110     if (get_temperature(i, &temp)) {
111       Serial.print(temp, 1);
112       Serial.println(" C");
```

```
113       }
114     }
115     Serial.println();
116
117     // Wait for a while, and then loop
118     delay(5000);
119   }
```