

Contents

1	Introduction	2
2	Les modules	2
2.1	Arduino nano	2
2.2	Écran LCD	2
2.2.1	Montage	2
2.2.2	Utilisation	2
2.3	Encodeur rotatif	3
2.3.1	Montage	3
2.3.2	Remarques	3
2.4	RTC	4
2.4.1	Montage	4
2.4.2	Utilisation	4
2.5	Carte SD	5
2.5.1	Montage	5
2.5.2	Utilisation	5
2.5.3	Remarques	6
2.6	Watchdog	6
2.7	Capteur PT1000	6
2.7.1	Informations	6
2.7.2	Mise en oeuvre	7
2.7.3	Remarques	8
2.8	Pompe	10
3	Projet complet	11
3.1	Fonctionnement	11
3.2	Wiring	11
3.3	Visuel	11

1 Introduction

Ce projet est une première version d'un prototype de système de gestion de chauffage. C'est une première ébauche d'une version plus complexe qui viendra plus tard. Ce prototype permet de mesurer la température à l'aide de quatre capteurs PT1000 et d'afficher ses température sur un écran. On a également rajouté la possibilité de sauvegarder ces température en fonction du temps sur une carte SD afin de pouvoir suivre leurs évolutions. Pour cela, on a besoin de :

- un arduino nano,
- un écran LCD,
- un encodeur rotatif (pour sélectionner le capteur dont on souhaite afficher la température),
- une RTC,
- un module carte SD,
- un wadtchdog,
- 4 capteur de température PT1000,
- une pompe.

2 Les modules

2.1 Arduino nano

Pour ce projet, on utilise un arduino nano, car c'est une carte compacte pour réduire la taille du montage final. Ensuite on va avoir besoin de beaucoup d'IO dans ce projet, un arduino ProMini sera insuffisant à ce niveau là.

2.2 Écran LCD

On utilise un écran LCD 16 caractères, 2 lignes (comme celui-ci). Ces écrans peuvent être utilisés avec un arduino grâce à la librairie LiquidCrystal.h (ou voir la page github) et on peut se contenter d'utiliser uniquement les pins de données D4 à D7 et laissé les pins D0 à D3 non connectées.

2.2.1 Montage

Le câblage de l'écran est présenté Figure 1, le schéma du module est présenté en vue de dessus les pins présentées non connectées seront reliées à l'arduino. Le potentiomètre sert à régler le rétro-éclairage de l'écran.

2.2.2 Utilisation

On appelle le constructeur:

```
LiquidCrystal lcd(RS,E,D4,D5,D6,D7);
```

en remplaçant RS, E, ... par les pins de l'arduino correspondantes. On peut branché l'écran aussi bien sur des pins logiques que numériques. Attention, il semblerait qu'on ne peut pas utiliser les pins A5 et supérieures (question de registre de la librairie).

Ensuite on initialise l'écran dans la fonction setup

```
lcd.begin(Nc,Nl);
```

où Nc est le nombre de caractères par ligne et Nl le nombre de ligne, dans notre cas on a Nc=16 et Nl=2. On peut écrit sur l'écran avec la fonction print(), se référer à la documentation de la librairie pour connaitre les autres fonctions.

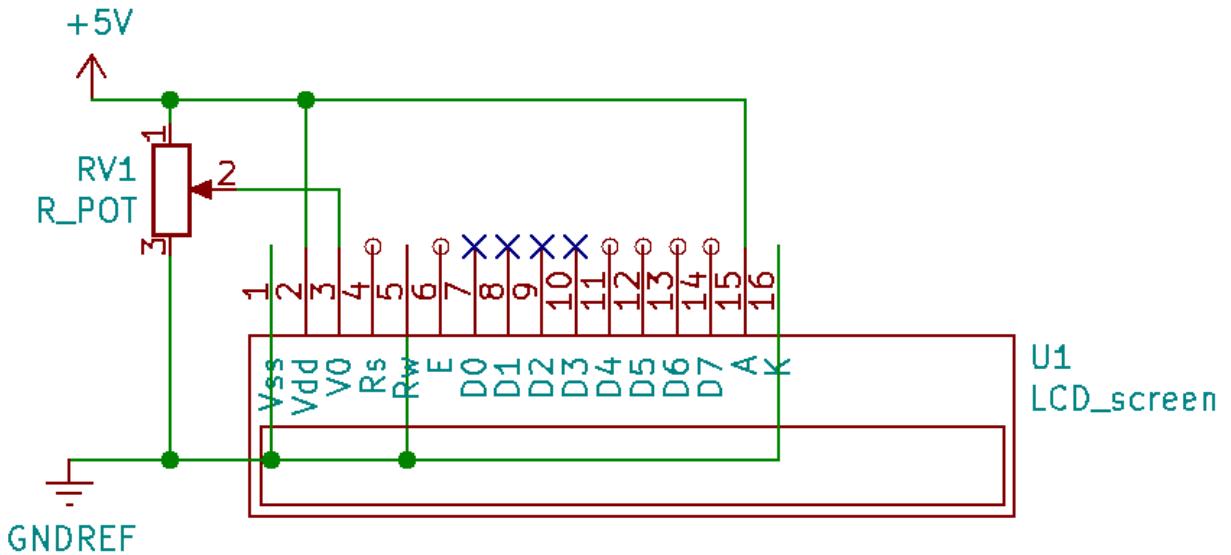


Figure 1: Montage de l'écran LCD.

2.3 Encodeur rotatif

On utilise un encodeur rotatif cliquable (comme celui-ci, un exemple d'utilisation de ce module peut se trouver ici. Dans le code on utilisera les fonctions présentées dans cette page pour déterminer la rotation du module.

2.3.1 Montage

Le câblage du module est présenté Figure 2. La pin SW sert à mesurer l'état de clique de l'encodeur, les deux autres servent à repérer la rotation de l'encodeur.

2.3.2 Remarques

Notons que pour le bon fonctionnement de ce module, il est préférable de brancher les pins DT et CLK sur les pins 2 et 3 de l'arduino nano, seules pins fonctionnant sur interruption.

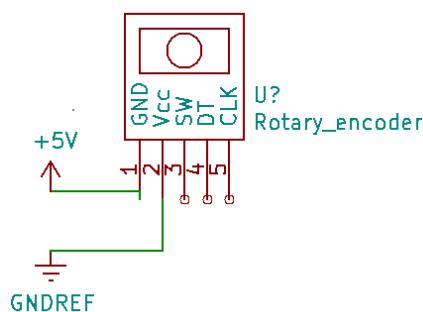


Figure 2: Montage de l'encodeur rotatif.

2.4 RTC

Une RTC (acronyme de Real Time Clock) est un module comprenant une horloge externe autonome de précision (module comprenant une batterie). Ce type de module permet de connaître l'heure exacte avec un microcontrôleur même s'il plante où s'il n'est plus alimenté. On utilise un module RTC DS3231 module compacte et peu cher, compatible 5V, basé sur le circuit intégré DS3231. Ce module communique en I2C, il doit donc être connecté aux pins A4 (SDA) et A5 (SCL) de l'arduino. Pour l'interfacer avec l'arduino, on utilise la librairie RTCLib.h (nécessite la librairie Wire.h).

2.4.1 Montage

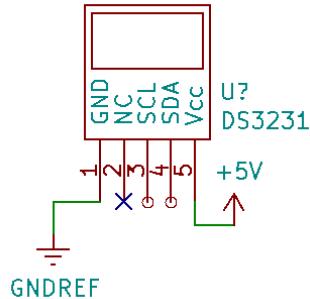


Figure 3: Montage de la RTC.

Le montage de la RTC est présenté Figure 3, elle est compatible 3.3 V et 5 V.

2.4.2 Utilisation

On inclus la librairie

```
#include<Wire.h>
#include<RTCLib.h>
```

On appelle le constructeur pour la bonne RTC (cette librairie gère plusieurs RTCs)

```
RTC_DS3231 rtc;
```

Ensuite on initialise la rtc dans la fonction setup

```
#ifdef AVR
    Wire.begin();
#else
    Wire1.begin();
#endif
rtc.begin();
```

On récupère un objet DateTime avec la fonction now() qui pourra permettre d'afficher la date et l'heure.

On peut également régler la date et l'heure de la RTC dans la fonction setup grâce à la commande

```
rtc.adjust(DateTime(__DATE__,__TIME__));
```

cette fonction permet de régler la RTC à la date et l'heure à la date et l'heure à laquelle le code a été compilé. Attention si on laisse le code tel quel, à chaque fois que l'arduino va démarer, la rtc va être remise à la date et l'heure de compilation, donc si l'arduino redémarre pour une raison ou pour une autre, la rtc sera déréglée. Une bonne manière de procéder consiste à brancher la RTC, compiler un code pour réfléchir l'heure, une fois fait, on coupe l'alimentation du circuit. On commente ou on efface du code la ligne de réglage puis on reflash l'arduino (la RTC doit être déconnectée du circuit). La RTC peut alors être rebranchée, elle restera bien réglée.

2.5 Carte SD

On utilise un module carte micro SD qui communique en SPI. Pour se servir de ce module, on utilise la librairie SD.h (nécessitant la librairie SPI.h). Avec un arduino, la communication se fait avec les pins 11 (MOSI), 12 (MISO) et 13 (CLK), le ChipSelect peut se faire avec la pin qu'on veut.

2.5.1 Montage

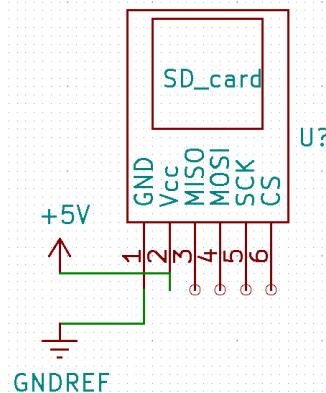


Figure 4: Montage du lecteur de carte SD.

Le montage du module de lecteur de carte SD est présenté sur la Figure 4. On ne montre pas ici les connections avec un arduino. Le brochage est présenté en vu du dessus. Le module est compatible 3.3 V et 5V.

2.5.2 Utilisation

On inclus la librairie

```
#include <SPI.h>
#include <SD.h>
```

On initialise le module dans la fonction setup

```
SD.begin(chipSelect);
```

On peut ouvrir un fichier myFile en écriture avec

```
myFile = SD.open(file_name, FILE_WRITE);
```

Ensuite on peut écrire dedans avec

```
myFile.print();
```

Comme pour l'écriture sur un lien série on peut utiliser la fonction println() pour insérer un caractère de fin de ligne après ce qu'on écrit dans le fichier. Il est nécessaire d'appeler la fonction

```
myFile.close();
```

pour écrire sur la carte SD. Tant qu'on ne l'appelle pas, ce qu'on veut écrire est stocké temporairement dans un buffer.

Si la carte SD est retirée, il faut clore la connection avec le module avec

```
SD.end();
```

2.5.3 Remarques

La communication SPI est sensible et a tendance à ne pas fonctionner avec des branchements breadboard. Durant mon premier essaie, je n'ai pas pu communiquer avec le module en utilisant une breadboard, pour mon premier test fructueux, j'ai connecté le module à l'arduino directement avec des fils femelle/femelle.

La librairie SD.h ne supporte pas des noms de fichier de plus de 12 caractères, 8 pour le nom et 4 pour l'extension (.txt ou .log par exemple).

Le retour des fonctions n'étaient pas bonne sur la première version de la librairie que j'ai utilisé, donc on ne savait pas si la carte était connectée ou pas. En mettant à jour la librairie SD.h en version 1.2.4, le retour d'erreur de la fonction open() peut permettre de savoir si la carte SD est bien présente dans le lecteur.

Attention à la carte SD qu'on utilise. La première carte utilisée semblait marcher correctement, on pouvait enregistrer un certain nombre de données mais rapidement la fonction SD.open() rentraitait un dossier vide qui semblait dire qu'il n'y ai plus de carte SD dans le lecteur. De plus, certaines fois où l'enregistrement ne semblait pas planter quand on ouvrait la carte SD sur ordinateur beaucoup de fichier apparaissaient avec des noms remplis de caractères ASCII "non classiques". Ces caractères pouvaient également apparaître uniquement dans les fichiers. Après différents tests, il s'est avéré au final qu'en changeant la carte SD ces problèmes ont été résolu.

2.6 Watchdog

Un watchdog est un système qui permet de redémarrer un microcontrôleur lorsque celui-ci plante. On peut se passer de l'utilisation d'un watchdog pour un petit projet qui va tourner quelques minutes par-ci-par-là, mais il est absolument indispensable pour un projet qui doit rester actif plusieurs heures, jours ou plus. Ici le projet doit pouvoir tourner plusieurs jours/semaines pour cette version, mais il devra durer tourner plusieurs mois dans cette version finale.

L'arduino possède un timer watchdog interne qu'on peut utiliser avec la librairie avr/wdt.h. Dans la fonction setup on initialise le timer avec la fonction

```
wdt_enable(WDTO_8S)
```

ici on règle le timer sur 8 secondes, mais il peut être réglé entre 15 millisecondes et 8 secondes. Si on ne réinitialise pas le timer avec la fonction

```
wdt_reset()
```

avant son terme, le watchdog redémarrera l'arduino.

Attention cependant lorsqu'on utilise un arduino nano, si l'arduino est un peu ancien et est flashé avec l'ancienne version de bootloader le timer est mal géré. Dans ce cas là si le watchdog s'enclenche, il n'est pas réinitialisé et donc au redémarrage il se réenclenche et donc l'arduino reboot en permanence. On peut le remarquer car la LED L clignote rapidement. Pour régler le problème, il faut utiliser un autre arduino en tant que programmeur ISP pour flasher la nouvelle version du bootloader (ou un autre bootloader compatible avec le watchdog).

Pour éviter des problèmes de redémarrage intempestif pouvant empêcher de reflasher l'arduino avec un nouveau code, il est conseillé de désactiver le watchdog dans la fonction setup en appelant

```
wdt_disable();
```

2.7 Capteur PT1000

2.7.1 Informations

Les capteurs PT1000 (famille des résistances de platine, dont le représentant le plus connu est le PT100) sont des résistances variables en fonction de la température. Un PTX possède une résistance de X Ohms à 0°C, et dont la résistance est donnée par :

$$R(t) = 1 + At + Bt^2, \quad t \geq 0^\circ C, \quad (1)$$

$$R(t) = 1 + At + Bt^2 + C(t - 100)t^3, \quad t < 0^\circ\text{C}, \quad (2)$$

avec $A = 3.9083 \cdot 10^{-3}$, $B = -5.775 \cdot 10^{-7}$, $C = -4.183 \cdot 10^{-12}$.

Les sondes seront utilisées pour mesurer la température de l'eau de chauffage, donc en condition normal d'utilisation, la température mesurée sera toujours comprise entre 0 et 100 °C. La résistance de la sonde en fonction de la température en considérant cette limite est montrée sur la Figure 5.

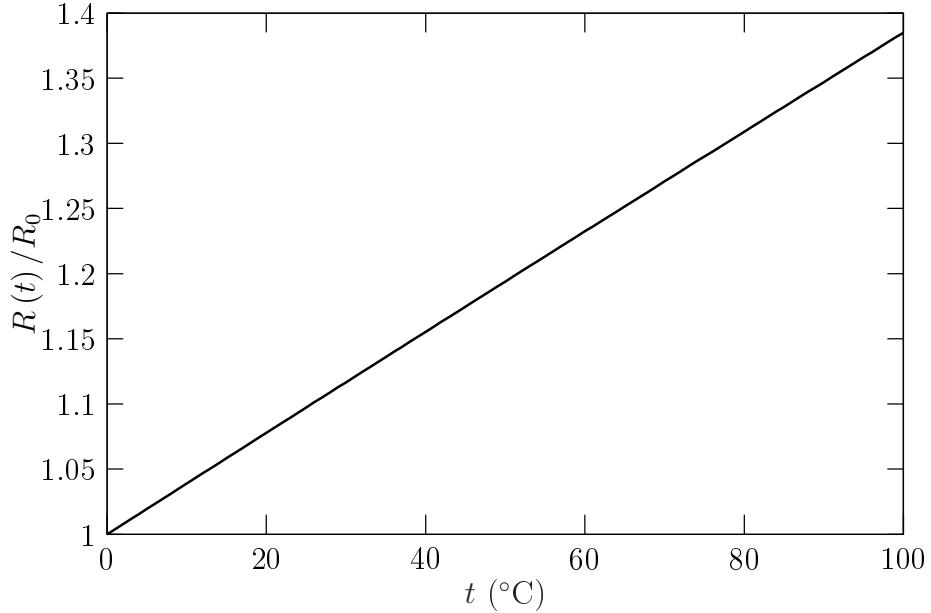


Figure 5: Variation de la résistance en fonction de la température.

On voit que la résistance normée varie entre 1 et 1.4 de manière quasi linéaire. Pour retrouver la température à partir de la résistance (en considérant des températures uniquement positives), on a juste à utiliser la formule de résolution d'une équation second degré, en sachant qu'on cherche la racine positive de l'équation :

$$t = \frac{-b + \sqrt{\delta}}{2a} = \frac{-A + \sqrt{A^2 - 4B(1-R)}}{2B}. \quad (3)$$

Si on considère une variation linéaire de la température ($R(t) = 1 + At$), alors la formule pour retrouver la température devient simplement :

$$t = (R - 1)/A. \quad (4)$$

L'erreur entre ces deux formulations est présentée sur la Figure 6, elle monte à environ 1.5 °C à 100 °C. L'erreur n'est pas très importante, surtout pour de faible température on peut se limiter à la formulation linéaire de la température. Mais pour des questions de précision (sachant qu'il faudra rajouter l'erreur de mesure de la tension) nous allons quand même utiliser la formule quadratique ici pour retrouver la température de la sonde.

2.7.2 Mise en oeuvre

On a vu que pour retrouver la température de la sonde, il faut pouvoir mesurer la résistance de cette sonde. Celà peut se faire de manière simple à l'aide d'un pont diviseur de tension en mesurant la tension aux bornes de la sonde, avec une résistance de référence R_r en série avec la sonde (*cf. Figure fig: pt1000 schematic*).

En procédant ainsi, la tension normalisée aux bornes de la sonde s'exprime :

$$\frac{V}{V_{cc}} = \frac{R(t)}{(R(t) + R_r)}. \quad (5)$$

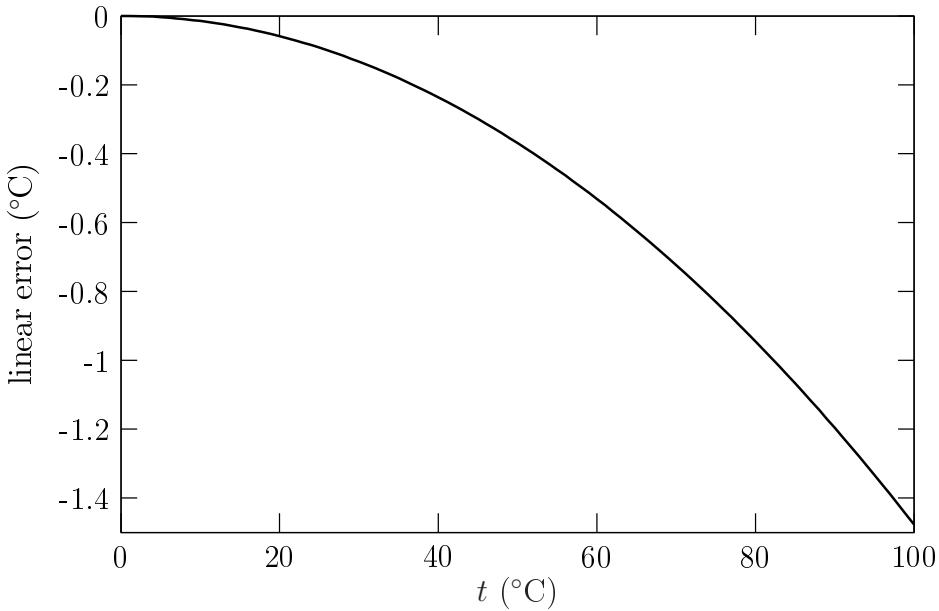


Figure 6: Erreur de l'approximation linéaire en fonction de la température.

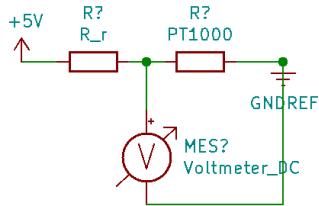


Figure 7: Montage d'un PT1000.

On peut déterminer la résistance de référence R_r permettant d'avoir la plus grande variation de tension aux bornes de la sonde. Pour cela on trace la variation de tension normalisée aux bornes de la sonde en fonction de la résistance de référence sur la Figure 8.

La résistance de référence optimum est de $1.18 \text{ k}\Omega$, avec cette résistance on a une variation de tension relative de 8.13%. Pour une résistance de référence comprise entre 920 et 1500Ω , on a une variation supérieure à 8%. Cette variation est faible, en considérant l'ADC 10 bits de l'arduino, cela se traduit par une variation de sortie de l'ADC de 80 points environ. Cette variation est tracé sur la Figure 9 en considérant une résistance de référence de $1 \text{ k}\Omega$.

On voit qu'on a une sensibilité de la mesure comprise entre 1 et 2 degré, en considérant l'erreur de l'ADC la précision de ce montage est de l'ordre de $+/- 3 \text{ }^{\circ}\text{C}$.

2.7.3 Remarques

Lors de la mise en place des sondes sur site, il y aura sûrement une rallonge entre les sondes et le système de mesure ces rallonges vont rajouter une résistance parasite en série de la sonde. Il sera préférable de les recalibrer sur site.

Pour améliorer la sensibilité de la mesure on pourrait utiliser un AOP en montage soustracteur (avec un LM358 par exemple). En choisissant bien la tension de référence et le coefficient d'amplification du montage, on peut gagner un facteur 10 de sensibilité.

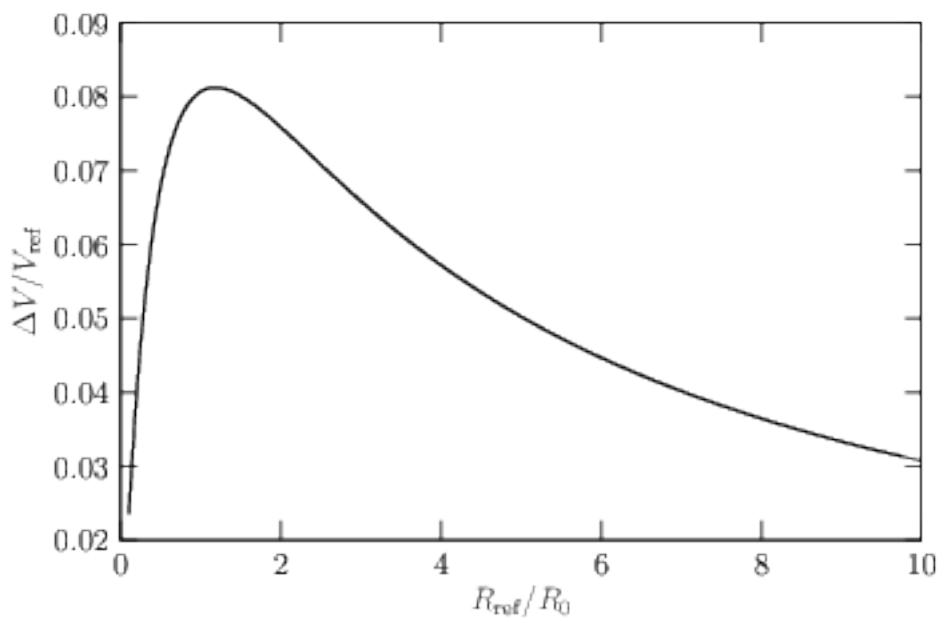


Figure 8: Variation de tension aux bornes d'un PT1000 (entre 0 et 100 °C) en fonction de résistance de référence.

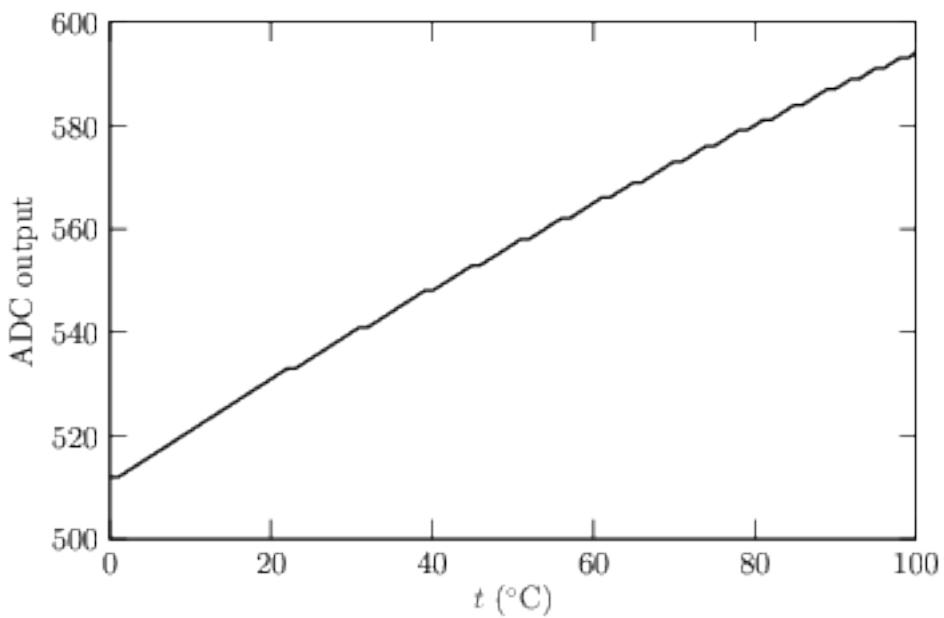
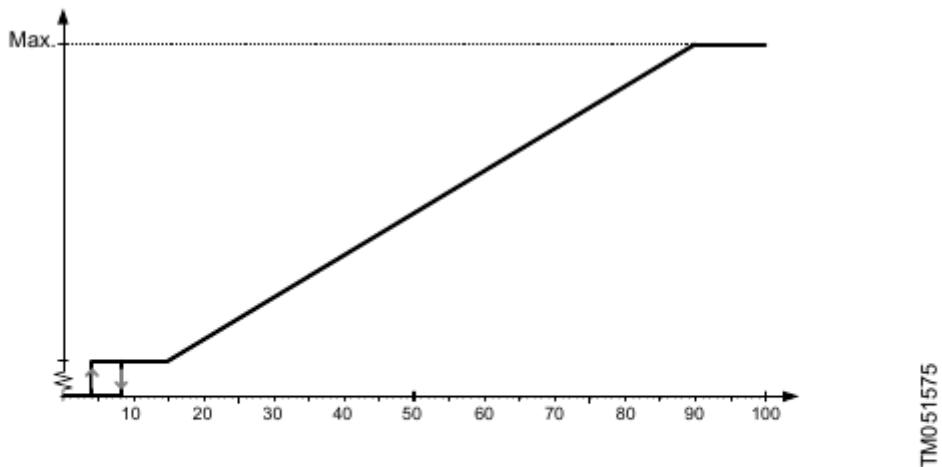


Figure 9: Sortie de l'ADC de l'arduino en fonction de la température.

2.8 Pompe

On utilise une pompe upm3 solar qui peut être contrôlée par PWM (tension comprise en 4 et 24 V). La définition des fils PWM de la pompe sont montrés page 52.

Le contrôle de la vitesse de la pompe en fonction de la PWM est donné par la courbe Figure 10 (page 18 de la documentation).



PWM input profile C (solar)

Axis	Value
X	PWM input signal [%]
Y	Speed
<hr/>	
PWM input signal [%]	Pump status
≤ 5	Standby mode: off
$> 5 / \leq 8$	Hysteresis area: on/off
$> 8 / \leq 15$	Minimum speed
$> 15 / \leq 90$	Variable speed from minimum to maximum speed.
$> 90 / \leq 100$	Maximum speed

Figure 10: Gestion de la pompe par PWM.

3 Projet complet

3.1 Fonctionnement

À l'aide de ce prototype, on affiche en "temps réel" l'heure (première ligne à gauche) et la température du capteur sélectionné T1 à T4 (deuxième ligne), la température affichée est la moyenne des dix dernières mesures. On peut changer le capteur à afficher avec l'encodeur rotatif. On affiche en haut à droite si une carte SD est présente ou non dans le lecteur de carte (attention, la réinitialisation du module SD est longue, environ 2 secondes, donc s'il n'y a pas de carte SD dans le lecteur l'affichage est saccadé). Si une carte SD est présente dans le lecteur, on enregistre les températures des quatre capteurs dans un fichier texte dont le nom est YYYYMMDD.txt tous les pas temps, par défaut ce pas de temps est de 5 minutes. En cliquant sur l'encodeur rotatif on peut changer le pas d'enregistrement de la température, on peut modifier ce pas de temps par pas de 10 secondes.

Le code de l'arduino est donné dans le dossier soft_arduino. Le firmware fonctionne de la façon suivant, à chaque boucle :

- on reset le watchdog,
- on demande l'heure à la RTC et on l'affiche sur l'écran lcd,
- on mesure la température pour les quatre capteurs et on la sauvegarde dans une "mémoire tournante",
- on affiche la température du capteur sélectionné (moyenné sur les 10 dernières mesures) sur l'écran lcd,
- si on a cliqué sur l'encodeur tournant on lance la fonction pour changer le pas d'enregistrement,
- si la carte SD n'était pas accessible à la boucle précédente on réinitialise le module SD,
- si la carte SD était accessible à la boucle précédente, on vérifie que c'est toujours le cas en ouvrant un fichier text (qui ne sert à rien sauf à vérifier que la carte est toujours accessible). Si on a une erreur c'est que la carte n'est plus accessible. On affiche sur l'écran qu'il n'y a plus de carte SD dans le lecteur,
- si l'heure donnée par la RTC est supérieur ou égale à l'heure du prochain enregistrement, on enregistre la température des 4 capteurs moyennée sur les 10 dernières mesures. Le nom du fichier texte est la date du jour sous la forme YYYYMMDD.txt.

3.2 Wiring

Le plan du projet complet est présenté Figure 11. Le projet est relativement simple mais utilise quasiment toutes les pins de l'arduino, seules les pins Rx et TX ne sont pas utilisées. Pour des questions de répétabilité et de fiabilité, il faut utiliser des résistances avec une bonne résolution, typiquement $\pm 1\%$.

Pour ne pas avoir un écran qui est allumé tout le temps, on pourrait rajouter un transistor ou un interrupteur sur l'anode (pin A) de l'écran.

Les capteurs T1 à T4 sont branchés respectivement sur les connecteur J2 à J5, soit respectivement les pins A7, A6, A3 et A2. La pompe est branchée sur le connecteur J1 (pin 9 de l'arduino), la masse de la PWM de la pompe est le fil bleu (*cf.* page 52 de la datasheet).

3.3 Visuel

On peut voir deux images du prototype complet monté sur les Figures 12 et 13.

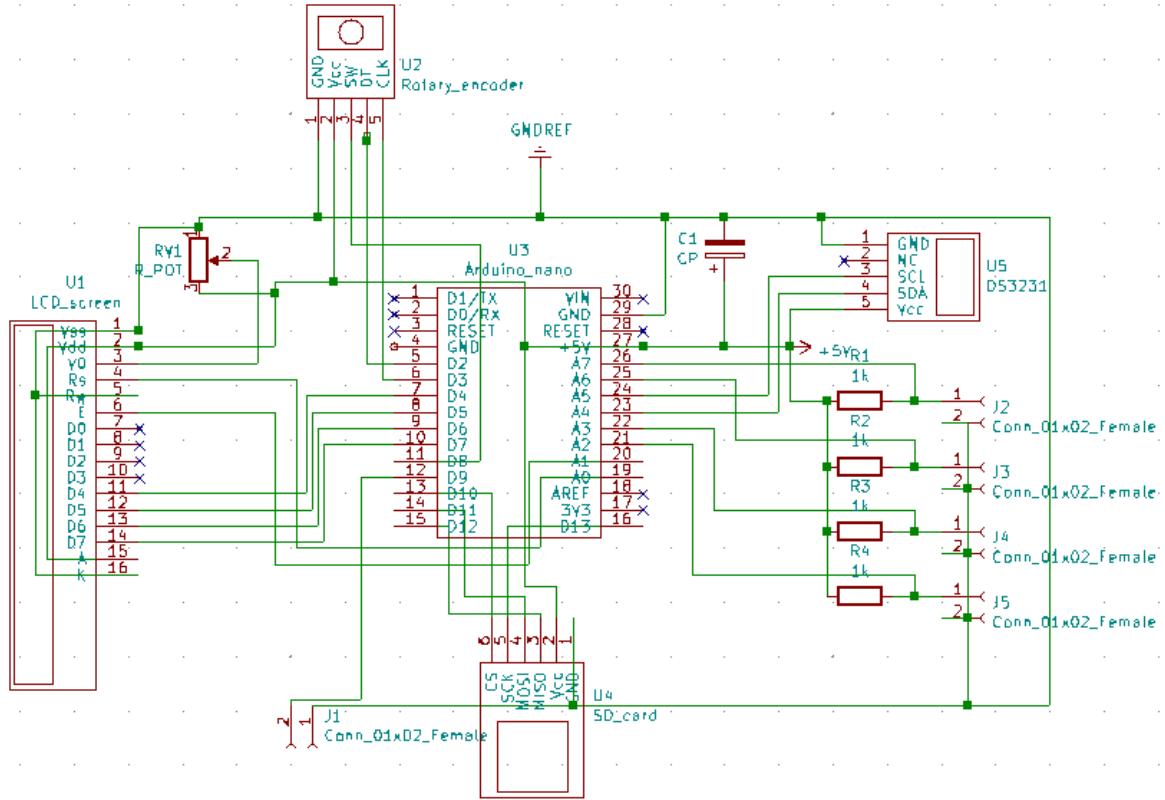


Figure 11: Plan du projet complet.

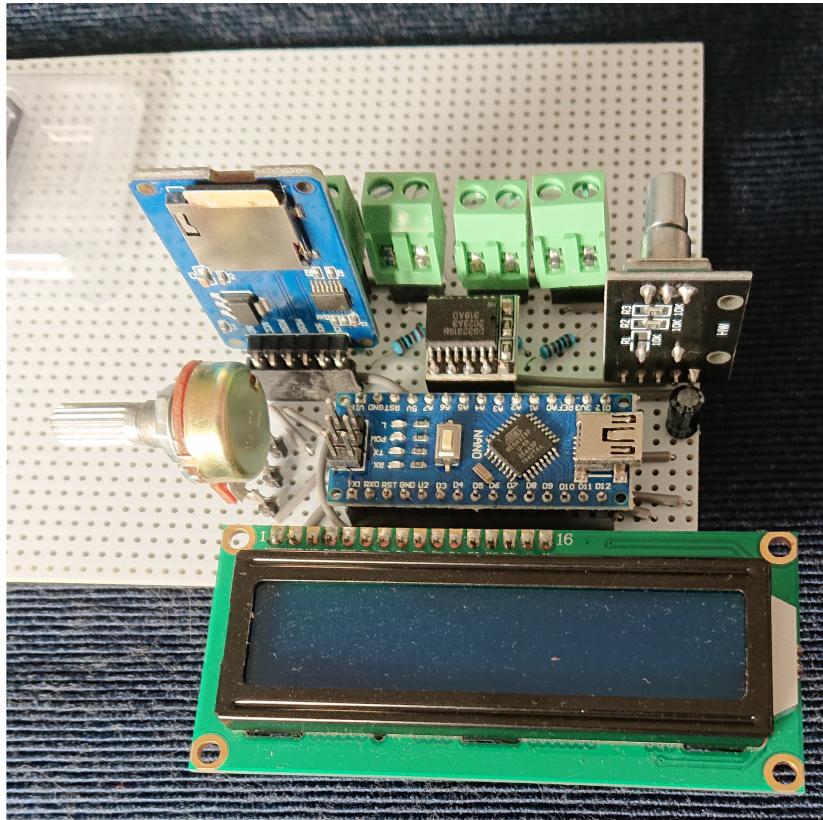


Figure 12: Photo du montage final.

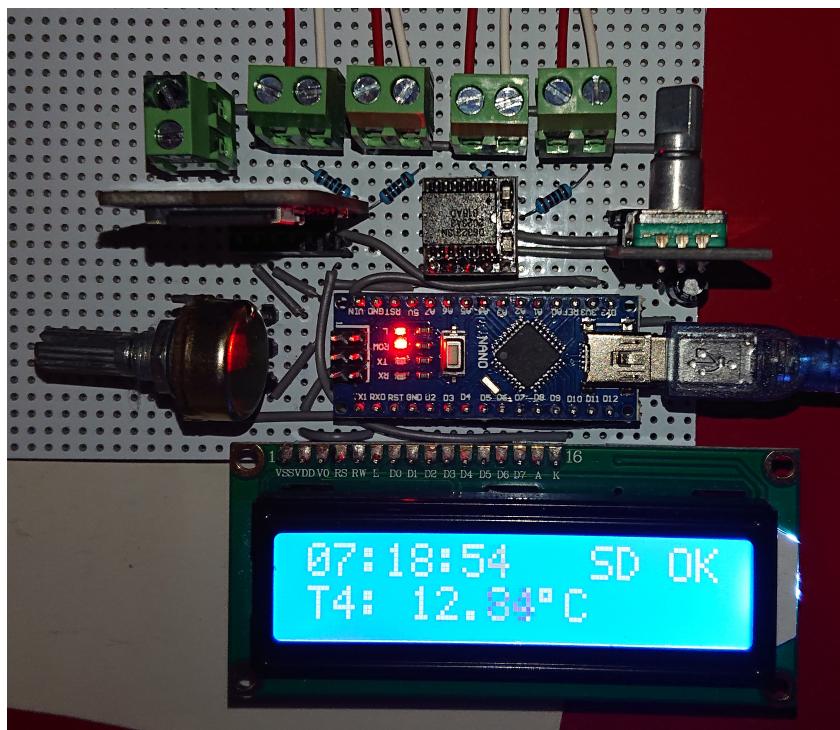


Figure 13: Photo du montage final.