Montage Arduino Nano

TEMPÉRATURE - HUMIDITÉ - PRESSION / ACCÉLÉRATIONS X-Y-Z

Jean-Paul Gendner F5BU

Tous ceux qui « bricolent » avec des plateformes de prototypage Arduino l'ont sans doute constaté comme moi : il est facile de trouver sur Internet des exemples de montages mettant en œuvre un timer, un bouton poussoir, tel ou tel convertisseur analogique-numérique, tel ou tel convertisseur numérique-analogique, etc., mais lorsque l'on souhaite utiliser simultanément deux ou plusieurs de ces éléments, les choses se compliquent passablement.

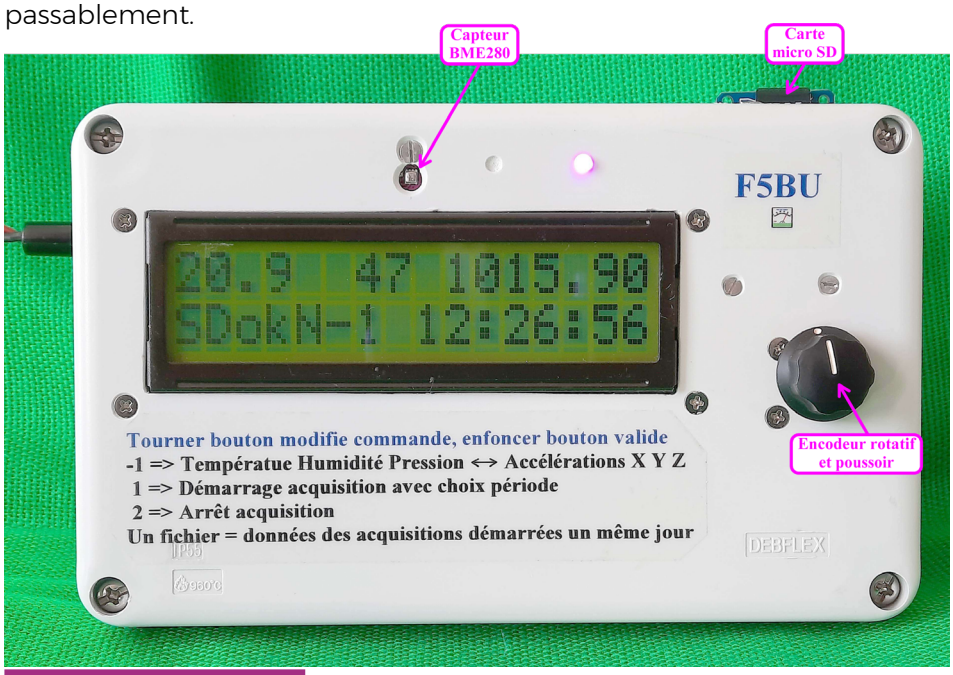

Figure 1 : La réalisation

Aussi, je me suis attelé à réaliser un montage qui soit une sorte de couteau suisse, basé sur un Arduino Nano, permettant l'utilisation simultanée et indépendante ou à la carte de différents éléments :

- des Serial.print[ln] faisant appel à la bibliothèque Arduino,
- un afficheur LCD parallèle (en utilisant la bibliothèque LiquidCrystal de Adafruit) ou d'un afficheur LCD I2C (en utilisant la bibliothèque LiquidCrystal_I2C de Adafruit),
- le convertisseur analogique-numérique de l'Arduino,
- un convertisseur analogique-numérique 24 bits ADS1220 dans différents modes (en utilisant la bibliothèque Protocentral_ADS1220 de Ashwin Whitchurch, un peu modifiée),
- un convertisseur numérique-analogique 12 bits MCP4821 (utilisant une bibliothèque MCP4821 créée en partant de la bibliothèque MCP48xx de Steve Gkountouvas),
- la bibliothèque SPI, utilisée pour le convertisseur 24 bits et le module SD et permettant d'ajouter d'autres accessoires SPI (mémoire, capteurs divers ...),
- l'EEPROM de l'Arduino (1024 octets pour le Nano),
- un module carte micro SD pour l'enregistrement des données, à travers un petit module perso d'adaptation à base de SN74LV1T125 (utilisant la bibliothèque SD de SparkFun Electronics),
- une horloge temps réel (RTC) gérée par I2C (en utilisant les bibliothèques wire et RTClib de JeeLabs),
- un encodeur rotatif avec interrupteur (avec du code basé sur un exemple de lastminuteengineers.com),
- le timer 1 ou 2 pour générer des interruptions,
- une DEL ...

Il y a quelques temps, dans le cadre des activités du REF-67, Olivier F4HTB a proposé une formation sur les Arduino Nano. Professionnellement, j'ai beaucoup utilisé les microcontrôleurs MSP430 de Texas Instrument pour des systèmes d'acquisition de données, miniaturisés, performants et de très, très, très faible consommation.

La programmation était faite en assembleur pour obtenir les meilleures performances. Une fois à la retraite, j'ai commencé à m'intéresser à d'autres microcontrôleurs. Alors cette formation tombait à pic pour acquérir un kit Arduino Nano et faire mes premier pas avec ce microcontrôleur qui m'était tout à fait inconnu. Pour commencer, j'ai réalisé un générateur de CW pour balise (voir Radio-REF 03-2021 p 12-15).

Plus récemment, je me suis lancé dans l'utilisation d'un composant qui m'intriguait depuis longtemps : un convertisseur analogique-numérique 24 bits. Il a été relativement facile de trouver sur Internet une bibliothèque ad-hoc, et d'obtenir des résultats tout à fait intéressants. J'ai alors voulu ajouter au montage un convertisseur numérique-analogique. Là aussi, les essais avec le fichier exemple fourni avec la bibliothèque ont été rapidement et facilement menés.

Comme je souhaitais, par exemple, utiliser le CNA pour générer une rampe en dents de scie durant des mesures de tensions avec le CAN, j'ai cherché à utiliser ces deux composants et un timer simultanément. Et là, cela a coincé. Chaque composant fonctionnait sans problème seul, mais pas les trois ensembles. Aussi, je me suis attelé à résoudre les problèmes, ce qui m'a, entre autres, amené à modifier certaines bibliothèques pour arriver au résultat souhaité.

Afin de pouvoir enregistrer des données, j'ai également ajouté un module carte micro SD et une horloge temps réel de manière à pouvoir dater les données acquises. Pour améliorer l'interface avec l'utilisateur, j'ai également ajouté un encodeur rotatif avec interrupteur. Arrivé à ce stade, j'ai été amené à mettre ce projet momentanément de côté pour réaliser des systèmes d'acquisition de données, un pour température, humidité et pression, l'autre pour les accélérations X, Y et Z.

Comme j'avais réalisé un circuit imprimé, et que j'en ai plusieurs exemplaires, j'ai trouvé pratique de l'utiliser, même si la plupart des éléments sont externes, car il comporte les principaux connecteurs pour les modules.

Le montage permet l'un ou l'autre type d'acquisition au choix.

Comme on peut s'en douter, tout tourne autour de l'Arduino Nano. Sont connectés pour cette réalisation :

- un LCD 16021 parallèle pour l'affichage sur 2 lignes de 16 caractères,
- un module BME280 pour les mesures de température, humidité et pression,
- un module ADXL335 pour les mesures d'accélération,
- un module carte micro SD pour l'enregistrement des données,
- un encodeur rotatif avec bouton poussoir pour l'interaction avec l'utilisateur,
- un module horloge temps réel RTC-DS3231.

Le schéma

Le schéma du montage est donné par la figure 2.

Les éléments grisés sont prévus pour être montés sur le circuit imprimé ou connectés à celui-ci, mais ils ne font pas partie du montage décrit, montage qui nécessite en fait peu de composants sur le circuit imprimé.

Remarques : les connexions nécessaires entre J45 et D10 et entre AREG et 3V3 ne sont pas représentées sur le schéma.

La résistance R28 est à choisir en fonction de la LED utilisée et de la luminosité souhaitée.

info

RÉDACTION D'ARTICLES POUR RADIO-REF : CONSEILS PRATIQUES

Afin de nous faciliter la tâche, nous vous demandons d'envoyer vos textes séparément, au format Word (ou Open Office) en joignant les photos à part, de préférence au format JPEG et de bonne définition.

L'ensemble est à adresser à radioref@r-e-f.org

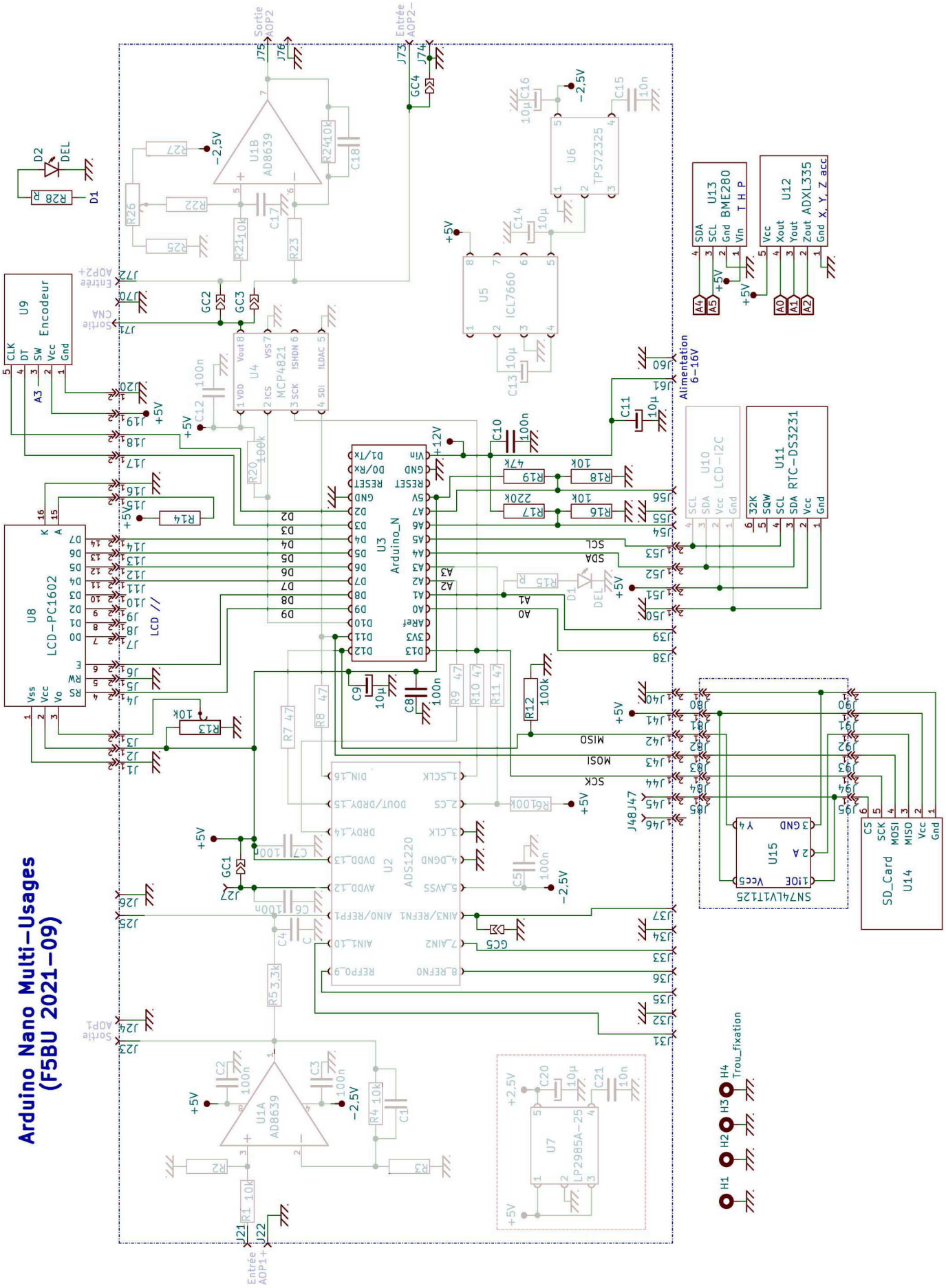

Figure 2 : Le schéma

Module LCD

Ajuster R13 pour le meilleur contraste et choisir R14 pour obtenir le rétroéclairage souhaité.

Il est bien sûr possible d'utiliser un module avec plus de lignes et/ou colonnes de caractères pour permettre l'affichage de plus d'informations, cela dit, il faut faire attention, car avec la possibilité de choisir entre les deux types d'acquisitions, 95 % de la mémoire programme sont utilisés et 71 % de la mémoire dynamique sur un Arduino Nano standard, 90 % de la mémoire programme après augmentation de celle-ci à 32 256 octets (voir annexe I).

Au lieu d'utiliser un LCD à interface parallèle, il est aussi possible d'utiliser un LCD I2C, ce qui libère 4 E/S du microcontrôleur, mais la bibliothèque pour ce LCD nécessite un peu plus de mémoire (environ 220 octets) ...

C'est la bibliothèque LiquidCrystal qui est utilisée.

Module BME280

Ce module est connecté à deux entrées analogiques de l'Arduino et fait appel aux bibliothèques Adafruit Sensor et Adafruit_BME280.

Module ADXL335

Ce module est connecté à trois entrées analogiques de l'Arduino et ne fait appel à aucune bibliothèque.

Pour améliorer l'exactitude des mesures d'accélération, il est conseillé d'effectuer une calibration du capteur. Voir Annexe II.

Module encodeur rotatif

Légère adaptation du code trouvé sur lastminuteengineers.com.

Module RTC-DS3231

L'horloge temps réel permet le maintien de la date et de l'heure durant les coupures d'alimentation de l'ensemble. Elle nécessite d'être initialisée. Pour cela :

- mettre en place une pile,
- mettre la valeur -1 à la place de 1 dans la ligne #define RTC_Present 1,
- comme indiqué dans le programme, modifier la ligne rtc.adjust(DateTime(2022, 1, 12, 20,40,00); avec des valeurs pour initialiser la date et l'heure,
- compiler puis télécharger le programme,
- appuyer sur le bouton à l'heure précise enregistrée dans la ligne rtc.adjust(...
- la date et l'heure s'affichent,
- remettre la valeur 1 dans la ligne #define RTC_ Present -1,
- compiler et télécharger à nouveau le programme.

Ce sont les bibliothèques Wire et RTClib qui sont utilisées.

Module carte micro SD

Ce module permet l'écriture et la lecture sur des cartes micro SD formatées.

Remarque : les modules en ma possession marchent très bien seuls, mais ils perturbaient dans certains cas la communication SPI avec d'autres éléments lorsqu'ils étaient branchés. Une étude du problème à l'oscilloscope a montré que le signal MISO n'était pas en haute impédance lorsque le module était inactif. Du coup, un petit montage avec un buffer 3 états a été inséré entre le module et l'Arduino.

Ce sont les bibliothèques SPI et SD qui sont utilisées.

Exemple de fichier généré :

'2021-12-29, 15:08:27, 13712, 5000, Temp
13, 22.67, 49.6, 994.41
4992, 22.70, 48.5, 994.39
9991, 22.72, 47.8, 994.37
...
'Fin=63968333

La première ligne comprend la date, l'heure de début, le nombre de millisecondes depuis le démarrage du système, la période d'acquisition en ms, le type d'acquisitions. Les lignes suivantes montrent le temps écoulé en ms depuis le début de l'acquisition, la température, l'humidité relative, la pression. L'enregistrement se termine par une ligne 'FIN= suivi du temps écoulé en ms.

La réalisation

La figure 3 donne l'implantation des différents éléments dans un boîtier PVC DebFlex (boîtier de dérivation pour l'électricité). Sur l'avant de l'appareil, une ouverture a été pratiquée en face du capteur BME280, alors que l'arrière de ce module a été recouvert de mousse polyuréthane pour l'isoler thermiquement de la chaleur dégagée par le reste de l'électronique.

Le temps de réponse pour les mesures de température est de l'ordre d'une quinzaine de minutes. Pour obtenir une réponse plus rapide, il faudrait sans doute placer le module entièrement à l'extérieur du boîtier, voire utiliser un petit capteur spécifique.

Les connexions des modules au circuit imprimé sont réalisées en grande partie à l'aide de prolongateurs pour permettre d'effectuer facilement des modifications.

La consommation est de l'ordre de 70 mA pour les mesures/enregistrements THP, un peu plus pour les mesures/enregistrements d'accélérations, surtout pour les grandes fréquences d'acquisitions.

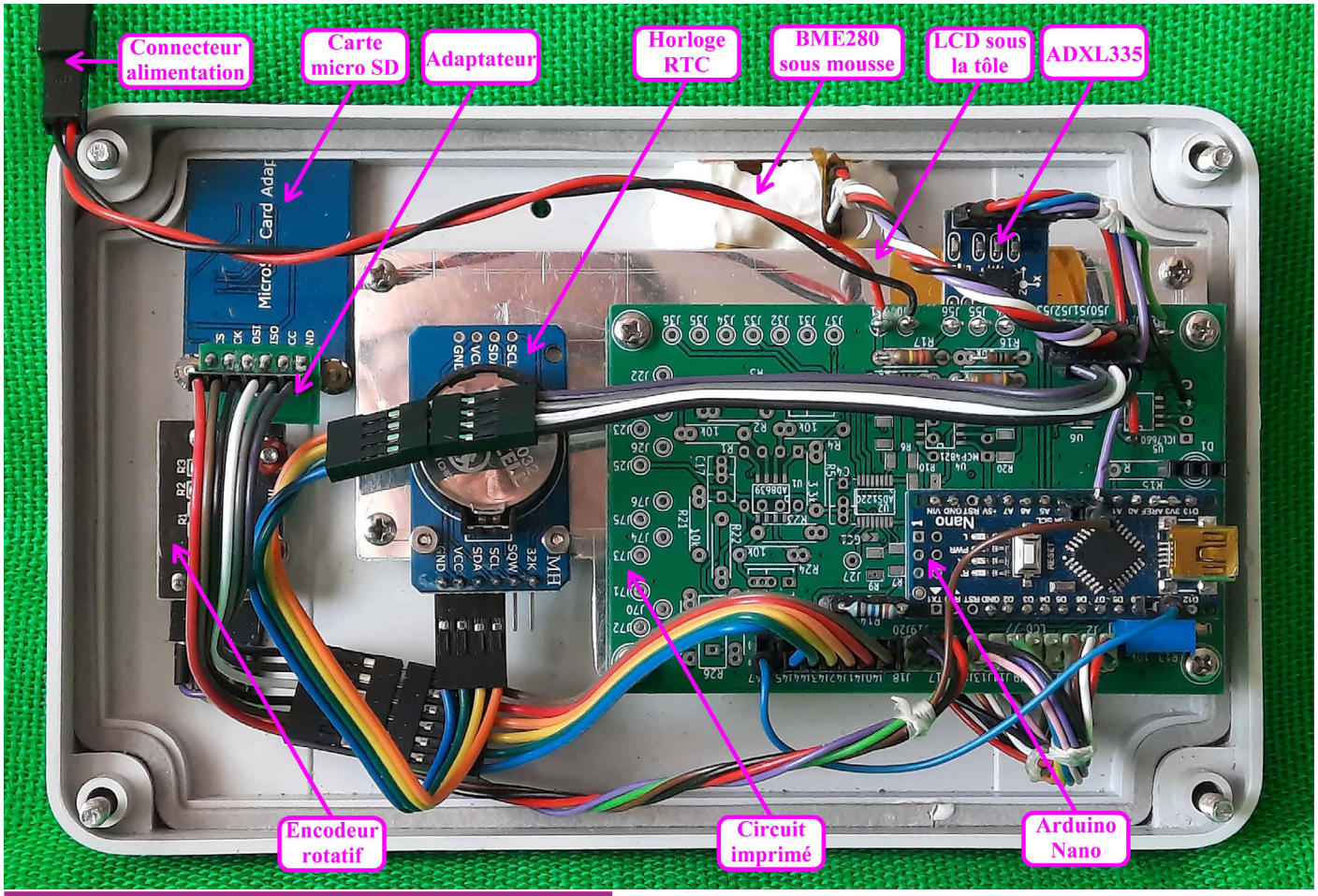

Figure 3 : L'implantation des différents éléments

Affichage à la mise sous tension

Ligne 1 : mon indicatif et la date.

Ligne 2 : la valeur de la tension 5 V qui alimente l'Arduino et, le cas échéant, la tension d'alimentation du montage.

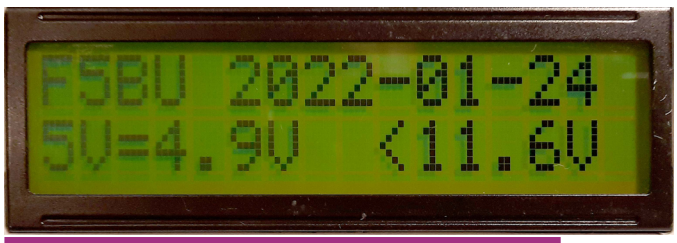

Figure 4 : Affichage à la mise sous tension

Affichage THP

Ligne 1 : température humidité pression

Remarque : la température est affichée en °C, l'humidité en % d'humidité relative et la pression en hPa, sans que cela soit précisé sur l'affichage, car ce LCD n'a que 16 caractères par ligne.

Il est toutefois possible de les afficher en ne mettant pas de valeurs décimales.

Ligne 2 : « SDok » indique que l'enregistrement sur la carte micro SD est possible, sinon, c'est « NoSD » qui est affiché ; le caractère suivant donne l'état du bouton poussoir « N » ou « P » pour enfoncé.

Le nombre qui suit indique le numéro pour une commande.

Sa valeur change en tournant le bouton rotatif.

Sur le côté droit on peut lire l'heure.

Figure 5 : Affichage température, humidité et pression

Affichage accélérations

Ligne 1 : accélérations axes X et Y en g

Ligne 2 : début de ligne, voir affichage THP, puis accélération axe Z en g

Figure 6 : Affichages Accélérations

Le programme

Le programme utilise le Timer 1 pour générer des interruptions toutes les 10 ms. Celles-ci mettent à jour une variable T10ms qui sert d'horloge.

Elle est déclarée en volatile uint32_t, c'est-à-dire en entier non signé de 32 bits à gérer en RAM et non dans un registre.

Cette variable peut prendre les valeurs de 0 à 4 294 967 295. Le débordement de ce compteur a donc lieu après plus de 497 jours.

Ceci n'est toutefois, a priori, pas une limite de la durée maximum d'acquisition, mais je ne l'ai pas testé !

Attention : le format non signé est utilisé parce que l'incrémentation d'une variable signée la fait devenir négative lorsque la valeur positive maximum est dépassée.

Mais le résultat d'une soustraction entre deux entiers non signés est toujours positif !

Cela complique passablement les tests, qui ne sont pas vraiment aussi intuitifs qu'avec des variables signées.

Le programme utilise les interruptions 0 et 1 pour prendre en compte toutes les actions sur l'encodeur rotatif en modifiant la valeur d'une variable Counter.

Cette variable est utilisée pour le numéro de la commande à exécuter et pour le choix de la période d'acquisition.

L'état du bouton poussoir est testé directement dans le programme.

Au début du programme, qui compte plus de 700 lignes, se trouvent des commentaires et les déclarations de toutes les constantes et variables.

Dans le Setup, se trouvent les initialisations du timer et de tous les modules.

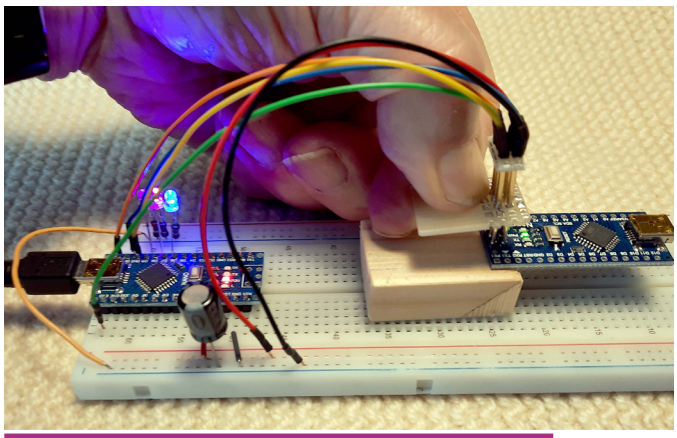

Figure 7 : Petit montage avec des pointes de touche à ressort permettant d'augmenter la mémoire d'un Arduino Nano sans effectuer de soudure

Dans la boucle (Loop) :

- dans le cas où une acquisition est en cours, on teste s'il est temps d'effectuer un enregistrement de données,
- on teste s'il est temps d'effectuer un affichage de données. Pour effectuer ces tests, on vérifie si T10ms est supérieur ou égal à la variable T10ms_Enr ou T10ms_Aff. Lorsqu'un test est vrai, la variable est mise à jour par addition respectivement de la période d'enregistrement choisie ou de celle d'affichage.
  Ensuite, les données sont lues et enregistrées sur la carte SD ou affichées sur le LCD.

Important : la variable T10ms étant mise à jour par interruption (100 fois par seconde), il est important d'arrêter les interruptions avant chaque utilisation de son contenu, sinon on risque d'utiliser une valeur erronée.

- on teste si le bouton poussoir est enfoncé.
  S'il est enfoncé, l'action dépend du numéro de commande affiché :
  - pour la valeur -1 on inverse le flag Acceleration qui définit le type de mesures THP ou Accélérations ;
  - pour la valeur 1 on affiche la plus petite période d'acquisition qui peut être modifiée.
    Cette valeur peut ensuite être modifiée par rotation du bouton puis est prise en compte lorsque le bouton est à nouveau enfoncé.
    Cette action démarre l'acquisition ;
  - pour la valeur 2, l'acquisition en cours est arrêtée ;
  - toute autre valeur est sans effet.

L'utilisation de la variable T10ms, mise à jour par interruption pour savoir s'il est temps d'effectuer telle ou telle action, garantit qu'il n'y a pas de dérive du temps à long terme.

Un fichier zip contenant le programme, le schéma et l'implantation du circuit imprimé peut être obtenu sur simple demande, par courriel indiquant le type d'application envisagée.

Remerciements

Grand merci à Olivier F4HTB et à Jean-Matthieu F5RCT, pour leurs conseils et leur soutien.

73, Jean-Paul F5BU (contact : f5bu@orange.fr)

Liens utiles

https://www.arduino.cc/

https://lastminuteengineers.com

https://www.youtube.com/watch?v=-MynHAz7TsI

Annexe I : Procédure pour augmenter la mémoire d'un Arduino Nano

C'est Olivier F4HTB qui m'a signalé l'existence d'une vidéo « Increase_Memory(.mp4) », par Al Williams, expliquant la procédure qui permet d'augmenter un peu la taille de la mémoire programme de l'Arduino Nano. Flirtant régulièrement avec la limite de cette mémoire, je me suis efforcé de comprendre, puis d'appliquer cette procédure, bien que la vidéo soit en anglais.

Principe : une partie de la mémoire programme est réservée pour le chargeur de démarrage du microcontrôleur (bootloader). Or, le chargeur utilisé actuellement sur la plupart des Arduino Nano est plus petit que l'espace réservé. Il faut donc modifier un « fusible » du microcontrôleur et un paramètre pour récupérer 1536 octets inutilisés. Ceci nécessite de reprogrammer le chargeur de démarrage en utilisant un deuxième Arduino.

Toute la procédure, utilisant un deuxième Arduino Nano, est décrite en détails dans le fichier Increase_Memory.pdf téléchargeable sur mon site f5bu.fr, dans l'onglet Téléchargements.

Annexe II : Procédure de calibration de l'ADXL335

- mettre la valeur 1 à la place de 0 dans la ligne #define AccCal 0,
- compiler et télécharger le programme,
- tourner le bouton pour afficher -1 et enfoncer le bouton pour passer en Accélérations,
- ce sont les accélérations qui s'affichent,
- toujours en position commande -1, enfoncer à nouveau le bouton,
- ce sont maintenant les 3 valeurs internes des capteurs qui s'affichent indéfiniment,
- rechercher, pour chaque axe, la valeur minimum et maximum statiques en modifiant l'orientation de l'appareil, en prenant garde à ne pas tenir compte des valeurs liées à des mouvements ou vibrations. Il est recommandé de toujours mettre l'appareil en appui sur un point fixe. Utiliser un miroir pour relever une des deux valeurs pour l'axe Z.
- une fois ces valeurs relevées, les utiliser dans les lignes définissant les valeurs d'offset (=(Vmax+Vmin)/2) et de sensibilité (=(Vmax-Vmin)/2) :

float Xoff = (629+419)/2 ;
float Xsens = (629-419)/2 ;
float Yoff = (625+412)/2 ;
float Ysens = (625-412)/2 ;
float Zoff = (635+425)/2 ;
float Zsens = (635-425)/2 ;

- mettre à nouveau la valeur 0 à la place de 1 dans la ligne #define AccCal 1,
- compiler et télécharger le programme.