TP/Projet d'introduction aux systèmes embarqués : réalisation d'une station météo

M. Madec[a,b], L. Werling[a,b], H. Omran[a,b], E. Dervieux[a,b,c], Th. Hingre[a], A. Phimant[a], M. Frey[a], W. Uhring[a,b]

[a] Télécom Physique Strasbourg, Université de Strasbourg, France
[b] Laboratoire ICube, UMR 7357, Université de Strasbourg / CNRS, France
[c] BioSENCY, Cesson-Sévigné, France
Pôle CNFM de Strasbourg MIGREST
Courriel : morgan.madec@unistra.fr

Cet article décrit un TP d'introduction aux systèmes embarqués réalisés avec les étudiants de Télécom Physique Strasbourg en deuxième année du cycle de formation ingénieur (BAC +4). Il vise à introduire les concepts fondamentaux des systèmes embarqués sur un exemple concret, en l'occurrence la réalisation d'une station météo, avec une approche de type projet. Les travaux pratiques se déroulent sur 2 jours (16 heures) pendant lesquels les étudiants apprennent à utiliser les fonctionnalités avancées du microcontrôleur (interruptions et timers), les protocoles de communication standards (SPI, I²C et 1-wire) et mettent en place une communication de type UART entre deux microprocesseurs.

I. Contexte

Télécom Physique Strasbourg forme des ingénieurs généralistes avec un spectre d'enseignements large allant de la physique fondamentale à l'informatique en passant par la photonique, l'électronique, l'automatique et le traitement du signal. Les étudiants de cette formation suivent un tronc commun de 3 semestres avant de se diriger dans un premier temps vers l'un des 4 départements de l'école lors du 4e semestre (*Physique, Informatique et Réseaux, Ingénierie des Signaux et des Systèmes,* et *Santé*), puis vers l'une des dix spécialités lors de la dernière année. L'une d'entre elles s'intitule *Électronique et Systèmes Embarqués* (ESE) et vise à former les étudiants plus spécifiquement à la conception et la programmation des systèmes embarqués, avec une emphase sur le volet *hardware* [1].

Au cours du 3e semestre, tous les étudiants suivent le module d'*Introduction aux Systèmes Embarqués*. Ce cours est précédé, en première année, d'un cours d'informatique en langage C, d'un cours d'électronique numérique et d'un cours d'introduction aux microcontrôleurs incluant notamment deux TP d'initiation à Arduino. Enfin, en préambule du TP/ projet, les étudiants suivent également un cours de 2 heures d'introduction générale aux systèmes embarqués et ont à leur disposition six vidéos de 20 minutes environ qu'ils sont invités à suivre en amont des TP et peuvent revoir pendant les TP [2].

Le choix de travailler avec Arduino est bien entendu discutable lorsque l'on parle d'une introduction aux systèmes embarqués, dans la mesure où il s'agit davantage d'une plateforme destinée au monde académique et au développement de petites applications plutôt que d'une technologie utilisée dans le monde industriel. Toutefois, nous avons fait ce choix afin de faciliter au maximum la prise en main de l'outil en occultant totalement certains aspects souvent difficiles comme la configuration du microcontrôleur et de son environnement de développement (IDE). Cela permet de répondre au mieux aux deux

Article publié par EDP Sciences et disponible sur le site https://www.j3ea.org ou https://doi.org/10.1051/j3ea/20221024

objectifs du module : *(i)* être suffisamment attractif pour donner aux étudiants les plus intéressés l'envie de poursuivre leur cursus vers la finalité ESE et *(ii)* être suffisamment général pour donner à tous les étudiants, quel que soit leur choix de spécialité, les clefs pour comprendre les problématiques liées aux systèmes embarqués et aborder la réalisation de tels systèmes dans leur domaine de spécialisation propre. Les étudiants de la spécialité ESE suivent par la suite d'autres modules plus en lien avec les attentes de l'industrie, par exemple le déploiement d'un OS temps réel sur un MSP430. De plus, le TP/projet se limite volontairement à des protocoles de communications filaires, les protocoles sans fils (Bluetooth, Zigbee, WiFi) n'étant abordés qu'en enseignement de spécialité.

II. Cahier des charges du projet réalisé

Les étudiants sont chargés de réaliser une station météo selon l'architecture décrite sur la Figure 1. Le système est composé de deux cartes Arduino communiquant entre elles via une liaison série (UART). La *carte-maître* gère essentiellement l'interface utilisateur, c'est-à-dire l'affichage des mesures sur un écran LCD et la commande du système par l'intermédiaire de deux boutons. Elle affiche en permanence (avec un rafraîchissement à chaque seconde) : l'heure au format *hh:mm:ss*, la valeur de température, ainsi qu'en alternance deux des 4 autres mesures météo (humidité, pression atmosphérique, qualité de l'air, luminosité). Le système est contrôlé par deux boutons permettant d'alterner entre les modes d'affichage d'une part, et de régler l'heure d'autre part. La *carte-esclave* pilote, quant à elle, les quatre capteurs via des protocoles de communication variés. La communication entre les deux cartes se fait en protocole série (UART). La *carte-maître* pilote l'ensemble des opérations : elle envoie une commande à la *carte-esclave* pour que celle-ci interroge ses capteurs uniquement quand cela est nécessaire, c'est-à-dire soit toutes les secondes, soit au moment de rafraîchir l'affichage après avoir appuyé sur les boutons.

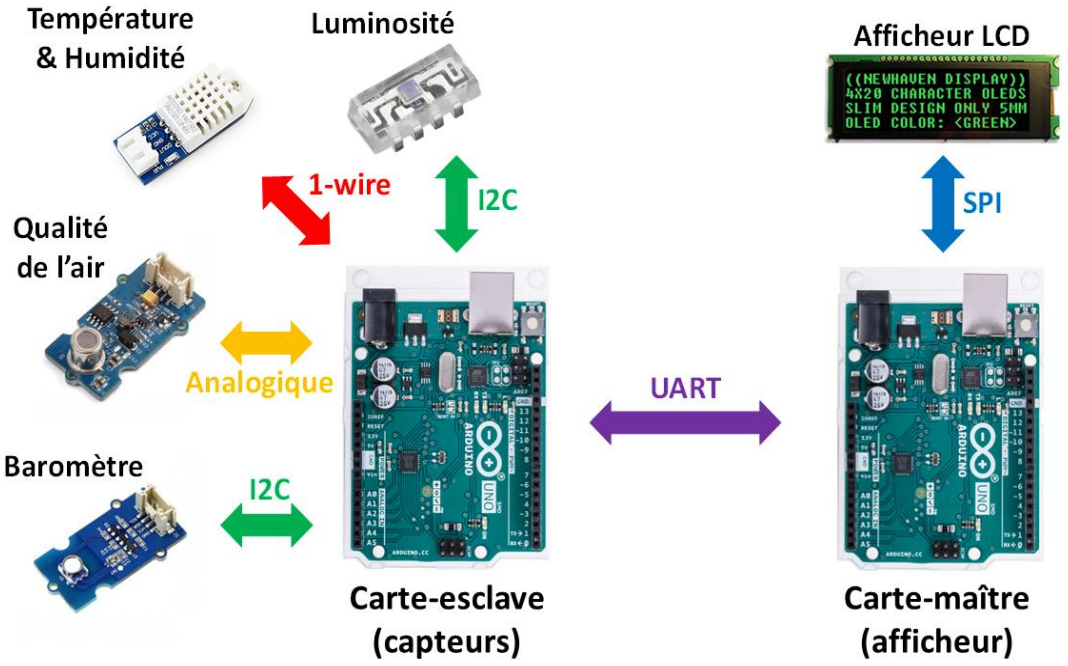

Fig.1. Architecture du système à réaliser. Il est constitué de deux cartes Arduino : la *carte-esclave* en charge du pilotage des capteurs (thermomètre, baromètre, hygrométrie, luminosité, qualité de l'air) au moyen de divers protocoles et la *carte-maître* qui gère l'interface utilisateur. Les deux cartes communiquent via une liaison série (UART).

III. Description du matériel

La carte-maître

La *carte-maître* est constituée d'une Arduino Uno surmontée d'une carte-fille réalisée par nos soins qui vient s'enficher sur l'Arduino (*shield*). Cette carte-fille contient un écran LCD 4x20 caractère Newhaven NHD-0420CW-AY3 (3) et deux boutons poussoirs connectés respectivement sur les entrées d'interruption INT0 et INT1 de l'Arduino (pattes 2 et 3 de l'Arduino Uno). Le schéma électronique de la carte et son rendu visuel une fois soudé sont donnés sur les Figures 2 et 3.

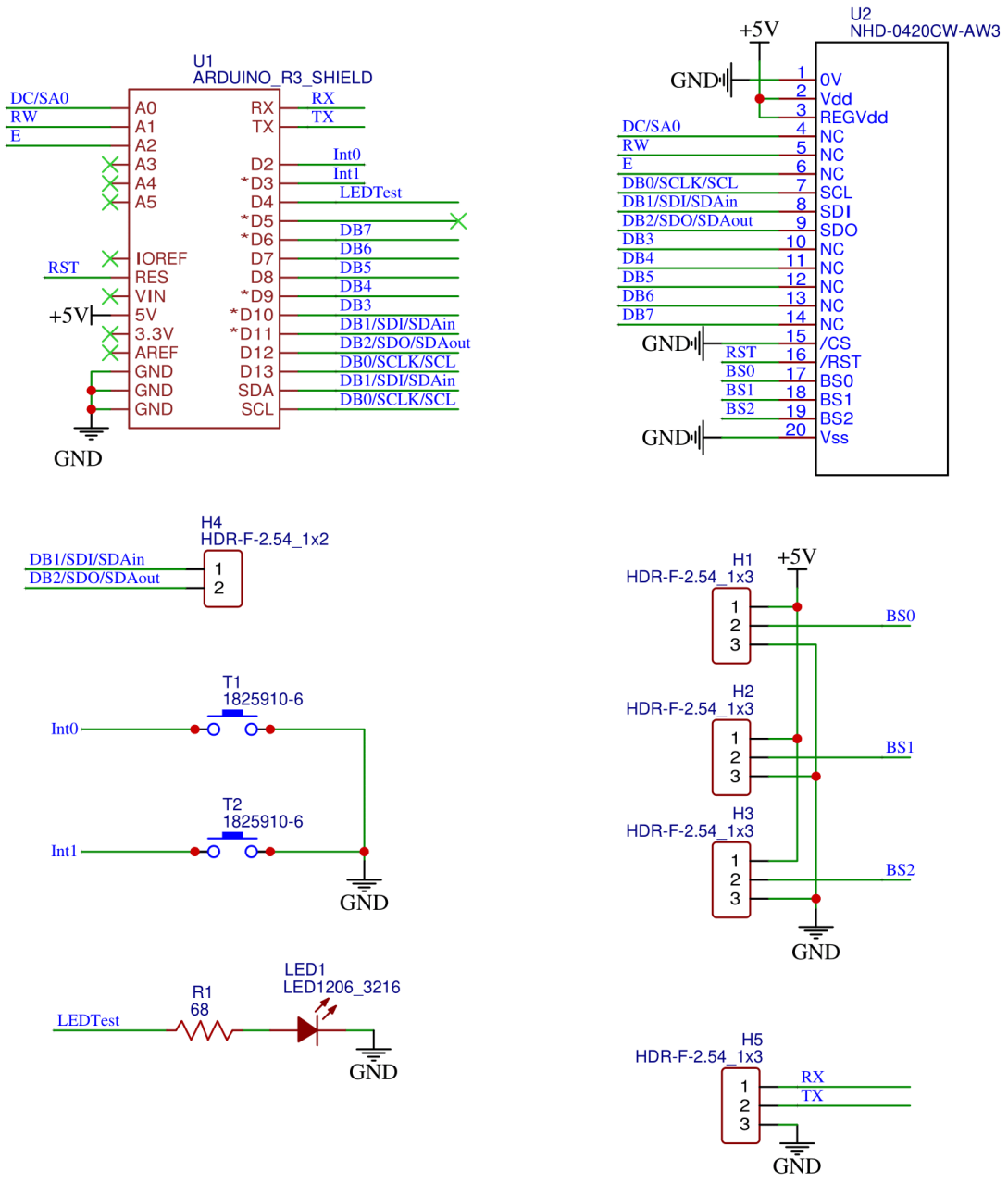

Fig.2. Schéma électronique de la carte-fille contenant l'écran LCD. Une version PDF de ce schéma est jointe en annexe du présent article.

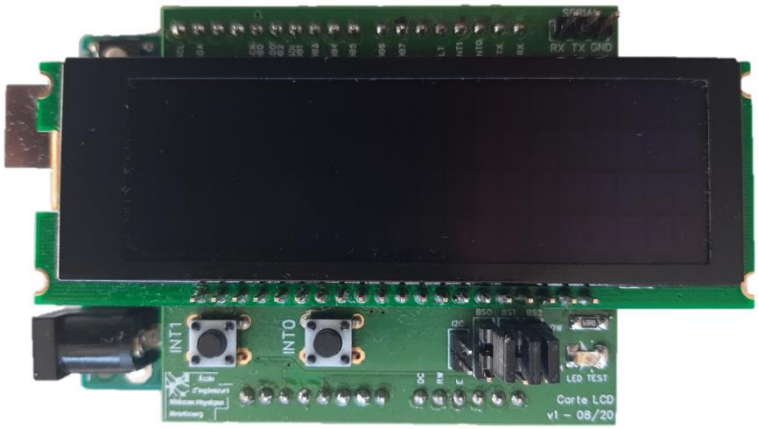

Fig.3. Carte-fille Arduino réalisée pour la *carte-maître* et composée d'un afficheur, de deux boutons-poussoirs de commande branchés sur les entrées d'interruption de l'Arduino et d'un connecteur UART.

L'écran LCD dispose nativement de 4 interfaces de communication avec le microcontrôleur : les interfaces parallèles 4 et 8 fils ainsi que les interfaces séries SPI et I²C. La carte-fille a été conçue pour permettre d'utiliser n'importe laquelle de ses interfaces en positionnant quatre cavaliers : trois d'entre eux mettent à 0 ou à 1 les entrées BS2, BS1 et BS0 permettant justement de configurer le mode de communication du contrôleur de l'écran (voir documentation [3]) tandis que le quatrième permet de connecter entre elles les pattes 8 et 9 de l'écran LCD pour former le signal bidirectionnel SDA du bus I²C. Pour le TP, nous incitons les étudiants à utiliser plutôt l'interface SPI. Ils pourront cependant tenter de communiquer avec l'écran en utilisant les autres interfaces s'ils en ont le souhait et/ou le temps. La carte dispose également de trois broches pour la communication UART (signaux RX, TX et GND) avec la *carte-esclave*.

La carte-esclave

La *carte-esclave* est également constituée d'une Arduino Uno et d'une carte-fille qui embarque les quatre capteurs de la station météo : un capteur de température et d'humidité, un capteur de pression atmosphérique, un capteur qualité de l'air et un capteur de luminosité. Le schéma électronique de la carte et son rendu visuel une fois soudé sont donnés sur les Figures 4 et 5.

Le capteur de température et d'humidité est un DHT22 (4) monté sur un module Grove. Il communique ses données via une liaison 1-wire. À chaque requête, initié par le microcontrôleur en plaçant la ligne de communication au niveau bas, le capteur délivre une trame de 40 bits à un baudrate fixe. Ces 40 bits sont constitués de 16 bits indiquant la température en dixième de degrés Celsius, 16 bits indiquant l'humidité en dixième de % et 8 bits de somme de contrôle.

Le capteur de pression atmosphérique est un HP206C (5) également monté sur un module Grove. Celui-ci communique la pression atmosphérique en pascals via un entier codé sur 20 bits et transmis en I²C. Il effectue également une mesure de température qui peut donc être corrélée à celle du DHT22 ;

Le capteur de luminosité est le VEML7700 (6) retournant une mesure d'éclairement ambiant également sous la forme d'un entier codé sur 16 bits. L'interface I²C permet la communication de cette valeur, mais également le réglage de la sensibilité et du temps d'intégration du capteur. Le facteur de gain entre la valeur binaire transmise et

l'éclairement réel en lux dépend de ces deux réglages. La table est donnée dans la documentation technique du capteur.

Le capteur de qualité de l'air est le Winsen MP503 (7) disposant d'une sortie analogique dont la tension est relative à la concentration de divers polluants dans l'air tels que les émissions d'alcool, le méthane ou encore la fumée. Dans le cadre de ce projet, on se contente d'afficher une valeur entière convertie via les convertisseurs analogiques numériques 10-bits de l'Arduino.

La carte dispose également du même connecteur permettant la communication série entre les deux cartes. Cette communication série est susceptible d'interférer avec la programmation des microcontrôleurs qui utilise également l'UART. Un interrupteur a donc été ajouté sur la carte pour permettre de rompre facilement la connexion entre les deux cartes pendant la phase de programmation, sans avoir à débrancher le montage. Enfin, la carte dispose de quatre connecteurs BNC permettant de visualiser facilement à l'oscilloscope les signaux I²C, le signal 1-wire issu du capteur de température/humidité et la tension analogique issue du capteur de pollution.

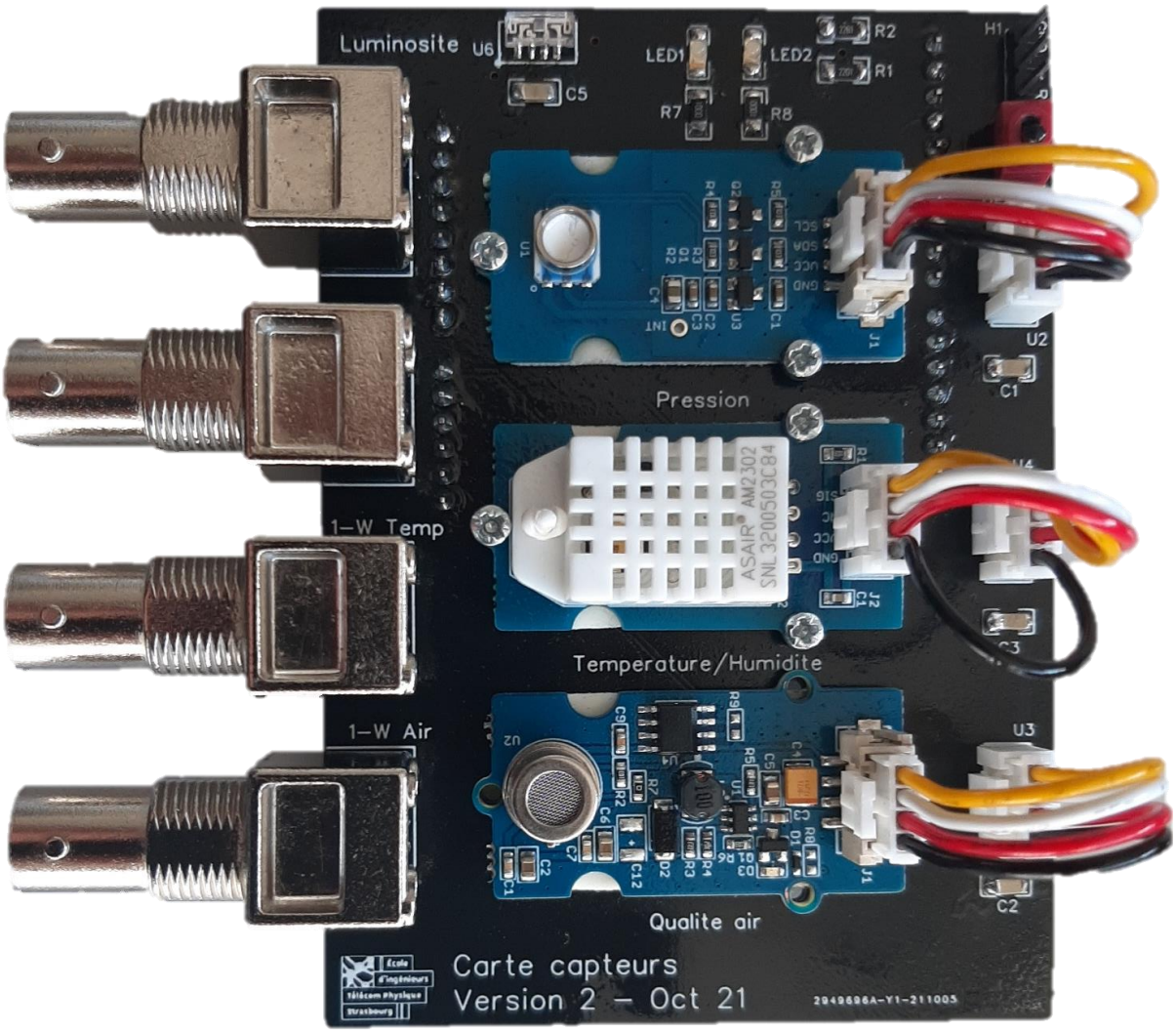

Fig.4. Schéma électronique de la carte-fille ; la *carte-esclave* et composée de quatre capteurs, quatre BNC pour visualiser les signaux issus des capteurs et un connecteur UART.

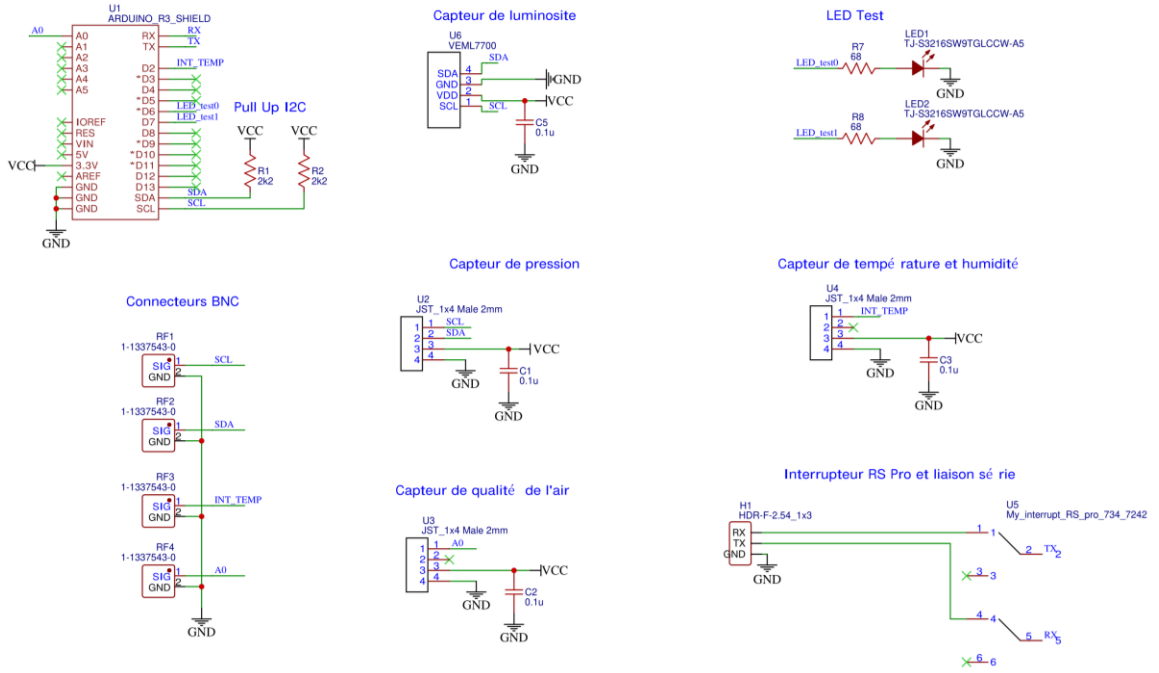

Fig.5. Schéma de la carte électronique de la carte-fille contenant l'écran LCD. Une version PDF de ce schéma est jointe en annexe du présent article.

IV. Contenu des cours magistraux et des vidéocours

La partie théorique de ce cours dure quatre heures réparties comme suit : deux heures de cours magistraux en amphi d'une part, et deux heures sous forme de vidéocours à suivre avant le TP et pouvant être revisionnée durant le TP d'autre part.

Le cours magistral est une introduction générale aux systèmes embarqués. Il est composé de quatre parties :

1. La première partie définit les concepts fondamentaux liés aux systèmes embarqués, et en particulier à leurs architectures matérielle et logicielle. Pour l'aspect logiciel, les différentes couches de programmation sont décrites (assembleur, langage C, système d'exploitation, middleware, application, etc.). La partie matérielle consiste en un bref rappel du cours de première année sur l'architecture d'un microcontrôleur.
2. La deuxième partie décrit d'un point de vue matériel et logiciel le fonctionnement d'un périphérique d'entrée-sortie. Le mécanisme d'interruption, passé sous silence en première année, y est notamment détaillé.
3. La troisième partie concerne la gestion du temps dans un système embarqué. Les différences fondamentales entre les délais bloquants, l'utilisation des timers ou encore l'implémentation d'une horloge temps réel (RTC) sont expliquées.
4. La quatrième partie est une description des principales caractéristiques associées à un protocole de communication : série ou parallèle, données brutes ou trames encapsulées, communication *half-duplex* ou *full-duplex*, communication point à point ou bus de communication, notion de débit, filaire ou sans-fils, etc. …

Les vidéocours sont plus spécifiques à l'environnement Arduino et au projet. Les six vidéos, d'une vingtaine de minutes en moyenne, sont disponibles sur YouTube (2). Elles traitent notamment :

1. de l'implémentation des interruptions dans l'environnement Arduino, en utilisant les fonctionnalités spécifiques attachées aux entrées INT0 et INT1, mais aussi en utilisant le mécanisme général d'ISR (*Interrupt Sub-Routine*),

2. de l'utilisation des *timers* sur l'Arduino Uno, avec pour exemple le clignotement d'une LED à intervalle régulier réalisé à l'aide du *Timer 2*,
3. du protocole de communication UART,
4. du protocole de communication I²C,
5. du protocole de communication SPI,
6. de quelques conseils pour démarrer le projet, notamment sur le bon usage des bibliothèques intégrées, la prudence à avoir lorsque l'on utilise des bibliothèques développées par d'autres utilisateurs, la notion de dépendance entre bibliothèques, l'utilisation de masques pour modifier spécifiquement un bit d'un registre et les bonnes pratiques lorsque l'on développe un projet conséquent (modularité, validation, gestion des versions et tests d'intégration).

V. Déroulement du TP

Le TP se déroule en mode projet, c'est-à-dire que l'on fournit aux étudiants le cahier des charges, les documentations techniques des 4 capteurs et les schémas des deux *shields*. Toutefois, pour les guider, nous leur proposons un certain nombre de points validation listés dans cette section. Le sujet de TP est joint en Annexe à cet article.

Pilotage de l'afficheur LCD

La première étape proposée aux étudiants est le pilotage de l'écran LCD. Pour cette partie nous avons opté pour une approche bas niveau en demandant aux étudiants de développer leur propre bibliothèque de fonction pour la communication avec l'écran, sans utiliser aucune bibliothèque existante à l'exception de *SPI.h*. La communication avec l'écran se fait par des trames de 24 bits composées de 15 bits fixes, un bit indiquant si on envoie une commande ou une donnée et 8 bits de commande ou de donnée, selon le protocole indiqué sur la Figure 6. Les étudiants sont invités à créer et tester les fonctions suivantes :

- envoi d'une commande codée sur 8 bits (validation à l'oscilloscope),
- envoi d'une donnée codée sur 8 bits (validation à l'oscilloscope),
- initialisation et configuration de l'écran LCD selon la procédure décrite dans la documentation du capteur (3),
- fonction d'effacement de l'afficheur,
- fonction de déplacement du curseur à une position donnée en argument.

Le point de validation de cette partie est de pouvoir afficher un message à un endroit spécifique de l'écran.

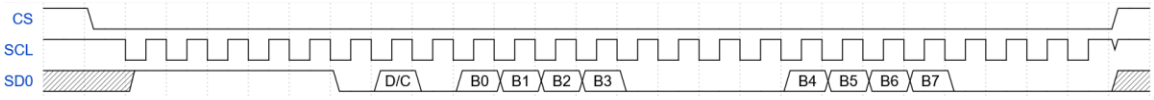

Fig.6. Chronogramme indiquant le protocole de communication SPI entre le microcontrôleur et l'écran LCD. Celui-ci est composé de trois trames consécutives de 8 bits chacune et permet d'envoyer à l'écran une commande ou une donnée codée sur 8 bits.

Gestion des interruptions et des timers

La deuxième étape du projet propose aux étudiants de tester individuellement les fonctionnalités du *Timer 2* du microcontrôleur ainsi que les entrées d'interruption interruptions INT0 et INT1 de l'Arduino. Le point de validation consiste ensuite à intégrer

ces trois fonctionnalités dans un programme affichant un message sur le moniteur série toutes les secondes, un autre à chaque fois que l'on appuie sur le bouton-poussoir relié à INT0 et un troisième lorsque l'on active l'autre bouton-poussoir.

Réalisation d'un chronomètre (intégration des deux premières parties)

Dans cette partie, les étudiants intègrent les fonctionnalités développées dans les deux premières parties au cœur d'une machine d'état. Le point de validation consiste à obtenir une horloge réglable affichant l'heure au format *hh*:*mm*:*ss* sur l'écran LCD. Le réglage de l'horloge se fait au moyen de deux boutons-poussoirs de manière assez classique (un bouton permet de passer mode réglage des heures, puis des minutes, puis des secondes tandis que le second permet de faire défiler les heures, minutes ou secondes pendant le réglage).

Communication avec les capteurs

La quatrième étape du projet est d'établir la communication entre la *carte-esclave* et ses quatre capteurs. Pour cela, les étudiants pourront utiliser les bibliothèques dédiées disponibles sur Arduino, à savoir :

- la bibliothèque générique *Adafruit_Unified_Sensor.h*,
- la bibliothèque *Adafruit_VEML7700.h* pour la communication avec le capteur de luminosité,
- la bibliothèque *HP20x_dev.h* pour la communication avec le capteur de pression atmosphérique,
- la bibliothèque *DHT.h* pour la communication avec le capteur de température et d'humidité.

Le capteur de qualité de l'air ne requiert quant à lui aucune bibliothèque particulière, car il suffit de lire sa sortie analogique avec le convertisseur analogique numérique du microcontrôleur.

Les fonctions sont validées individuellement à l'aide du moniteur série, puis intégrées en un seul et unique programme. Le point de validation est ici de répondre à l'envoi sur le moniteur série d'un caractère 'H', 'T', 'P', 'Q' ou 'L' par la valeur mesurée par le capteur correspondant (ex : température pour le caractère 'T', etc.). Enfin, à titre d'exercice, les étudiants sont invités à observer à l'oscilloscope les trames I^2C et 1-wire émis respectivement par les capteurs DHT22 et HP206C, à les décoder et à vérifier leur cohérence avec les informations présentes dans la documentation technique (4)(5).

Communication UART entre les Arduino

La cinquième étape du projet est d'établir la communication UART entre les deux cartes Arduino. Pour cela, nous proposons aux étudiants la réalisation de quelques fonctions simples : l'allumage d'une LED sur une carte lorsque l'on appuie sur un bouton de l'autre carte, la programmation d'un Arduino en mode écho (renvoi systématique sur la liaison série par un Arduino d'un caractère dès réception de celui-ci), ainsi que la transmission de chaîne de caractères, validés via le moniteur série.

Intégration de la station météo

La dernière étape du projet est l'intégration des différentes parties développées précédemment pour la réalisation d'un système complet répondant au cahier des charges imposé.

VI. Résultats attendus

Les résultats attendus sont décrits dans une vidéo disponible sur YouTube (8) et sur les figures 7 à 9. Sur la figure 7, on peut voir le montage complet, avec les deux cartes Arduino portant les deux cartes-filles. Les deux Arduino sont reliés au PC via deux ports USB distincts et sont reliés entre elles via leur liaison série (fils bleu et orange). Sur l'afficheur, la première ligne correspond à l'heure, la seconde à la température, puis les deux suivantes à la pression atmosphérique, l'humidité, la qualité de l'air et la luminosité alternativement. La figure 8 est un montage montrant les 4 affichages possibles.

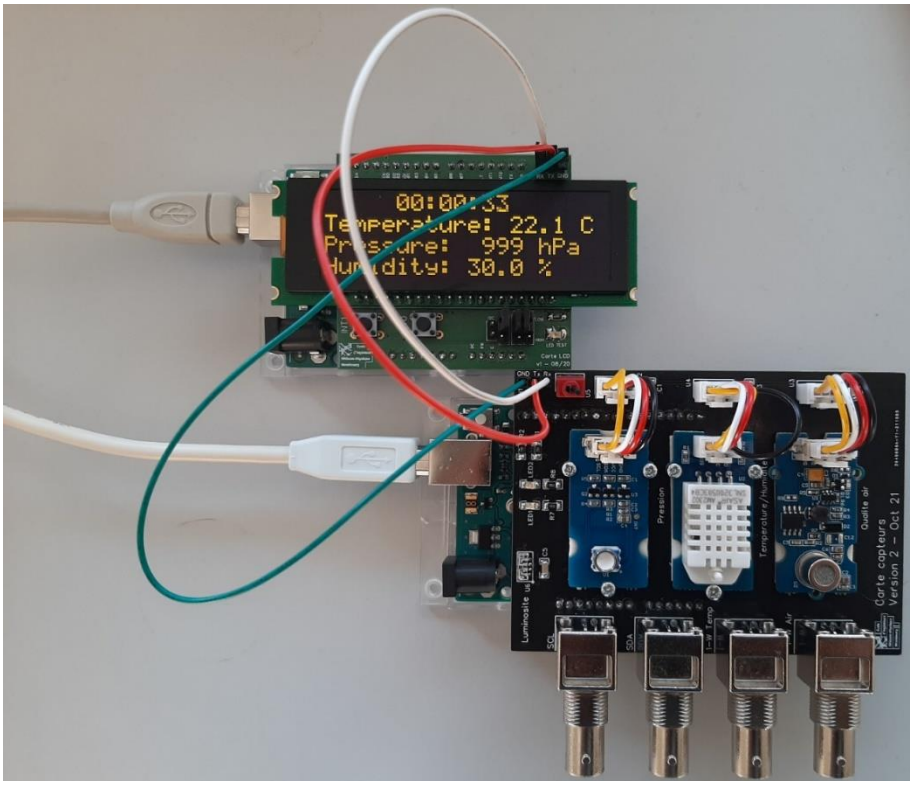

Fig.7. Montage complet composé des deux cartes Arduino surmontées des deux cartes-filles, connectées au PC via un port USB pour l'alimentation et la programmation et connectées entre elles via l'UART (RX/TX).

Fig.8. Montage montrant les quatre configurations d'affichage possibles. L'heure et la température sont affichées en permanence. La qualité de l'air, la luminosité, la pression atmosphérique et l'humidité quant à elles, sont affichées en une alternance contrôlée par un bouton-poussoir.

D’un point de vue de la programmation, deux IDE Arduino sont nécessaires, l’un pour piloter la carte maître et l’autre pour piloter la carte esclave (Figure 9). Pour chaque IDE, le moniteur série associé permets de suivre la communication qui s’établit entre les deux Arduino. On voit par exemple en bas à gauche de la figure 9 que le maître communique avec l’esclave au moyen de caractères (‘T’ pour demander une mesure de température, ‘P’ pour une mesure de pression, et ‘H’ pour une mesure d’humidité). De la même manière, on peut lire la réponse de l’esclave en bas à droite de la figure 9. Il s’agit d’une trame de 12 caractères. Les 4 premiers correspondent à la température (ici 24.4°C), les quatre suivants à la pression atmosphérique (1006 hPa) et les quatre derniers à l’humidité (38.4 %).

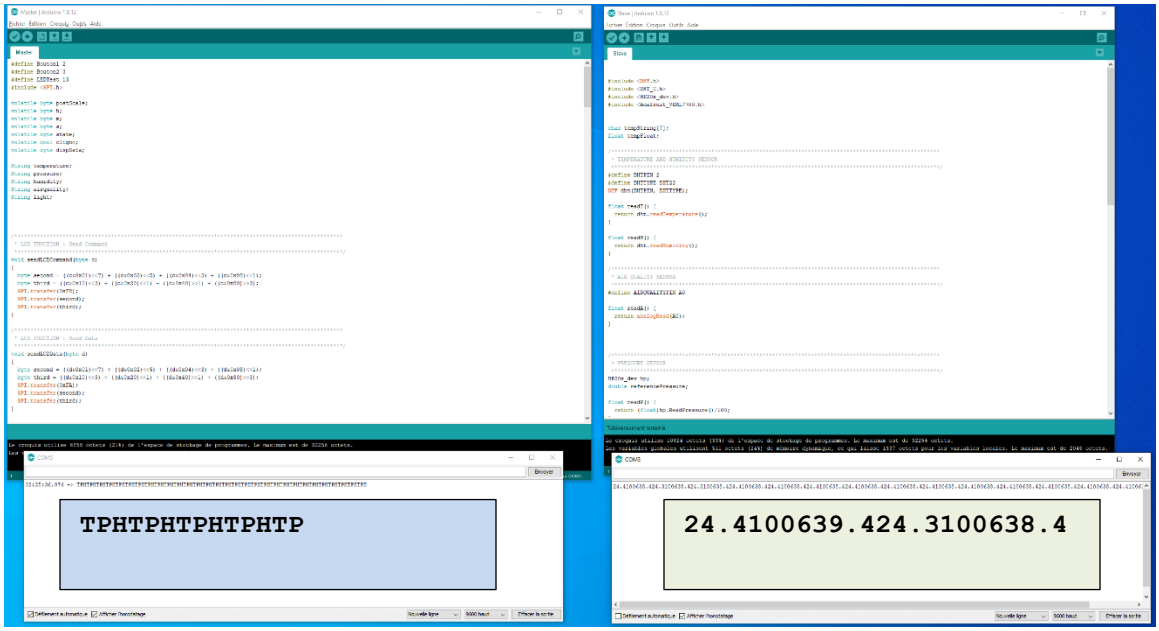

Fig.9. Capture d’écran montrant les deux IDE Arduino, celui du maître à gauche et celui de l’esclave à droite. Les fenêtres du bas correspondent aux moniteurs séries respectives de chacun des IDE. Les valeurs affichées dans les moniteurs séries ont été agrandies pour plus de lisibilité et repris dans les cadres bleu (maître) et vert (esclave).

VII. Déroulement des séances

Le TP a été réalisé pour la première fois cette année sur une promotion de 86 étudiants. L’ensemble des binômes a eu la satisfaction d’avoir fait fonctionner indépendamment une horloge réglable avec affichage de l’heure sur écran LCD d’une part, ainsi que la communication entre une Arduino et ses capteurs avec affichage des valeurs sur le moniteur série d’autre part. La communication entre les deux microcontrôleurs via UART a été plus problématique et a mis en avant la difficulté pour certains étudiants à concevoir un programme, même simple, lorsqu’aucune piste n’est suggérée. Finalement, environ 20% des binômes sont parvenus à la réalisation du système complet.

Le tableau I résume le temps moyen passé par les étudiants ayant terminé le projet sur chacune des parties. L’établissement de la communication avec l’afficheur LCD est, sans grande surprise, la partie la plus chronophage du TP. Elle nécessite une lecture et une compréhension approfondie de la documentation technique de l’afficheur et la réalisation de fonctions non triviales permettant par exemple de réaliser la trame 24-bit SPI à partir de la donnée 8-bit que l’on souhaite envoyer. De plus, la vérification et le débogage du code généré sont assez difficiles puisque dans la plupart des cas, si le code est erroné, le seul

comportement observable est le non-fonctionnement de l'afficheur, sans plus d'indice sur l'origine du problème. L'observation de la trame SPI permet d'identifier d'éventuelles erreurs dans les fonctions permettant l'envoi d'une commande ou d'une donnée, mais il est très difficile d'identifier des erreurs dans la séquence de démarrage.

TABLEAU I. Temps moyen passé par les étudiants pour la réalisation des différents sous-projets décrits dans la Section V. Le temps total alloué au projet complet étant de 16 heures.

Tâche	Temps moyen (h)
Pilotage de l'afficheur LCD	5
Gestion des interruptions et des timers	3
Réalisation d'un chronomètre	1
Communication avec les capteurs	2
Communication UART entre les Arduino	3
Intégration de la station météo	2

VIII. Conclusion

Le TP tel que nous l'avons conçu nous semble être un bon compromis entre un système suffisamment complexe pour susciter la réflexion et forcer les étudiants à adopter une démarche rigoureuse tout en utilisant une plateforme suffisamment simple (Arduino) pour ne pas perdre trop de temps sur la prise en main du microcontrôleur ou de son logiciel de programmation. La première partie sur l'afficheur permet aux étudiants de descendre très proche du matériel avec une étude approfondie de la documentation technique de l'afficheur et le développement de fonctions bas niveau. La deuxième partie sur les capteurs développe d'autres compétences liées à l'analyse et la réutilisation de fonctions existante. Enfin, l'intégration de l'ensemble des briques élémentaires dans une application complète permet aux étudiants de toucher de manière concrète à des problématiques propres au développement logiciel, telles que par exemple l'intégration de code, les tests unitaires, le débogage de systèmes complexes, la gestion des versions, etc.

À l'issue du TP, une enquête a été menée auprès des étudiants pour récolter leur avis. Celle-ci révèle que les étudiants ayant répondu à l'enquête ont majoritairement apprécié le TP. Le taux de satisfaction moyen est de 71,6% (Figure 10). 71% des étudiants ont apprécié le format du TP (2 jours bloqués) bien qu'un certain nombre d'étudiants ont trouvé les journées trop longues et ont formulé dans les commentaires libres d'autres alternatives qui méritent d'être étudiées (deux jours non consécutifs, trois journées, mais moins intenses en termes de nombre d'heures). La satisfaction des étudiants à propos du travail en autonomie est bonne (65,5 % de satisfaction moyenne). Enfin, plus de 80% des étudiants (moyenne globale à 70,3%) estiment avoir acquis des compétences grâce à ce TP. Les étudiants ont également été invités à se prononcer sur les vidéocours mis à leur disposition. La moitié d'entre eux jugent qu'elles préparent bien au TP même si certains soulèvent le fait qu'elles n'entrent pas assez dans le détail sur les protocoles de communication. Pour finir, parmi les autres critiques négatives formulées par les étudiants en commentaire libre, on trouve notamment les difficultés rencontrées sur la partie afficheur qui a tendance à décourager en début de TP.

Enfin, à la question de savoir quelles compétences les étudiants ont l'impression d'avoir acquises durant ce TP, les réponses majoritaires sont :

- la programmation sur Arduino d'une application complète et conséquente,

- l'utilisation de documentation technique pour comprendre le fonctionnement des composants électroniques utilisés, mais aussi des bibliothèques de fonctions intégrées,
- l'élaboration d'une démarche permettant d'arriver à ce but en décomposant le problème et en testant individuellement chaque partie.

Fig.10. Résultat de l'enquête menée à l'issue des TP. Le taux de participation est de 32,5 % (28 réponses sur 86 étudiants). Pour les quatre questions dont les réponses sont représentées sous la forme de camembert, le taux de satisfaction moyenne affichée en haut à droite est calculé en appliquant une pondération de 1 aux réponses très positives (vert foncé) ; 0,67 aux réponses plutôt positives (vert clair), 0,33 aux réponses plutôt négatives (orange) et 0 aux réponses très négatives (rouge). Pour la figure du bas, le pourcentage de réponse indiquée correspond au pourcentage d'étudiants, sur les 28 ayant répondu au sondage, qui estiment avoir acquis des compétences dans le domaine spécifié.

Annexes

Le présent article est complété par un certain nombre de ressources mis à disposition en ligne sur le site du journal J3EA :

- Annexe 1 : sujet de TP tel que transmis aux étudiants,
- Annexe 2 : schémas électroniques en version PDF de la carte-fille portant l'afficheur LCD,
- Annexe 3 : fichier de topologie en version PDF de la carte-fille portant l'afficheur LCD,
- Annexe 4 : fichiers de topologie au format Gerber de la carte-fille portant l'afficheur LCD,
- Annexe 5 : schémas électroniques en version PDF de la carte-fille portant les capteurs,
- Annexe 6 : fichier de topologie en version PDF de la carte-fille portant les capteurs,
- Annexe 7 : fichiers de topologie au format Gerber de la carte-fille portant les capteurs,
- Annexe 8 : code source pour le microcontrôleur de la carte-maître,
- Annexe 9 : code source pour le microcontrôleur de la carte-esclave.

Références

1. Télécom Physique Strasbourg – École d'ingénieur de l'Université de Strasbourg. *Site web: https://www.telecom-physique.fr/*
2. Chaîne YouTube contenant les vidéos de cours associés aux travaux pratiques décrits dans ce papier : https://www.youtube.com/playlist?list=PLSuadvVXceA2N5o7QT1mYIlVV7NpYWnKd3
3. Documentation technique de l'écran LCD Newhaven NHD-0420CW-AY3 disponible sur le site web de Newhaven. *Site web: https://www.newhavendisplay.com/specs/NHD-0420CW-AY3.pdf*
4. Documentation technique capteur de température et d'humidité DHT22 disponible sur le site web de Lextronic. *Site web: https://www.lextronic.fr/capteur-humidite-temperature-dht-22-14983.html*
5. Documentation technique capteur de pression atmosphérique HP206C disponible sur le site web de Lextronic. *Site web: https://www.lextronic.fr/module-grove-barometre-101020068-20868.html*
6. Documentation technique capteur de luminosité VEML7700 disponible sur le site web de Digikey. *Site web: https://www.digikey.fr/product-detail/fr/vishay-semiconductor-opto-division/VEML7700-TR/VEML7700CT-ND/5820244*
7. Documentation technique capteur de qualité de l'air MP503 disponible sur le site web de Winsen. *Site web: https://www.winsen-sensor.com/sensors/voc-sensor/mp503.html*
8. Vidéo de démonstration : https://youtu.be/Fn9i2dyTzvA