Comment aborder l'élevage de précision dans l'enseignement agricole ? : approche à partir d'exemples

Thibault Maillot, L'Institut Agro Dijon

PRÉSENTATION DE LA PARTIE 1 : Matériel pédagogique

Table des matières

1. TD : Notions de capteurs et élevage de précision

« *En matière d'élevage, la précision s'applique à chaque animal. L'enjeu est de rassembler des informations correspondant à chacune des unités d'un troupeau. Ces informations portent sur l'état génétique (potentiellement par génotypage lorsque les coûts auront diminué), sur les caractéristiques physiques, le comportement physiologique, l'état sanitaire et les performances productives. Ces informations permettront de mieux anticiper les décisions relatives à chaque animal et optimiser la conduite générale du troupeau.* »

(Association internationale pour une agriculture Écologiquement Intensive)

La puce (ou tag) RFID (*Radio Frequency IDentification*) comporte une puce électronique et une antenne. Cette dernière est utilisée pour fournir de l'énergie au tag et pour transmettre des données entre la puce et une station de lecture.
Ces transferts s'effectuent généralement à courte distance (de quelques centimètres jusqu'à un mètre). Cette distance de transfert peut être augmentée en ajoutant une batterie au tag RFID [8].
Dans le domaine de l'élevage, les puces RFID sont majoritairement utilisées pour permettre l'identification des animaux. Cette identification est utilisée afin de faciliter la pesée, le tri et la gestion des troupeaux, via des matériels et logiciels tiers.

L'utilisation de cette méthode d'identification peut être étendue par l'ajout d'un autre capteur (e.g. un accéléromètre) afin d'effectuer un suivi sanitaire et physiologique des animaux. Ce suivi est effectué grâce à des comparaisons entre les mesures de chaque animal et des résultats théoriques issus de modèles prédictifs. En fonction des résultats de comparaison, un logiciel d'aide à la prise de décision va indiquer à l'éleveur si un animal a besoin d'une attention particulière. La Figure 1 schématise le concept d'élevage de précision, utilisant une puce RFID pour identifier un animal [2].

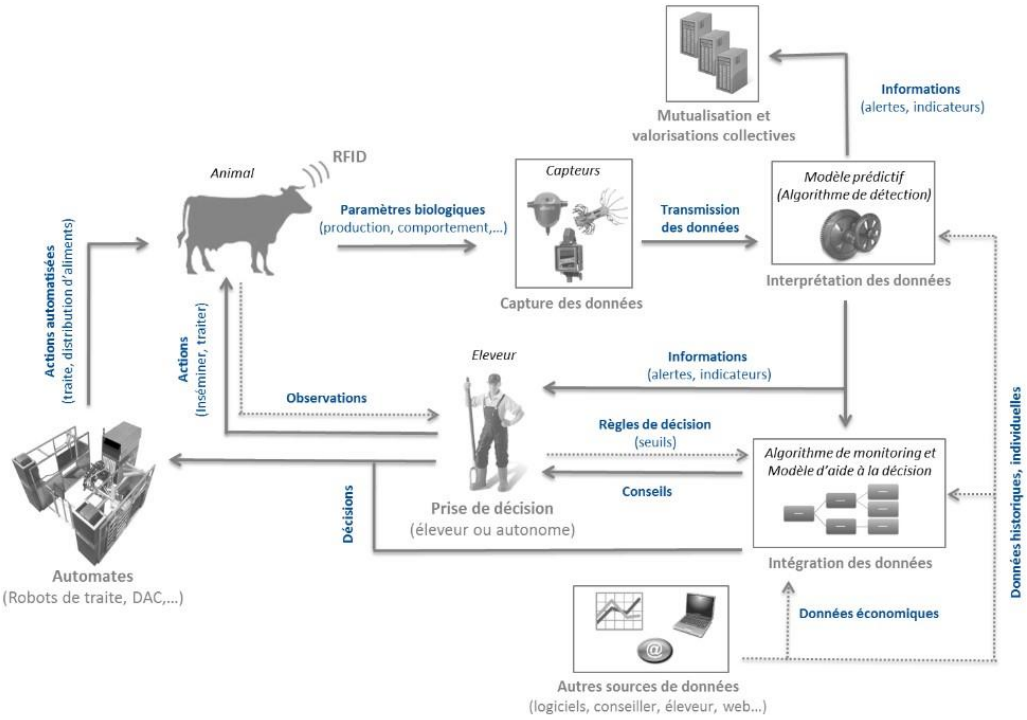


Figure 1 : Concept d'élevage de précision

SOURCE : C. ALLAIN, A. CHANVALLON, P. CLEMENT, R. GUATTEO et N. BAREILLE, « Élevage de précision : périmètre, applications et perspectives en élevage bovin », in *Rencontres autour des Recherches sur les Ruminants (21e éd.)*, p. 3-10, Institut de l'Élevage-Inra, 2014. [2] © idele-Droits réservés

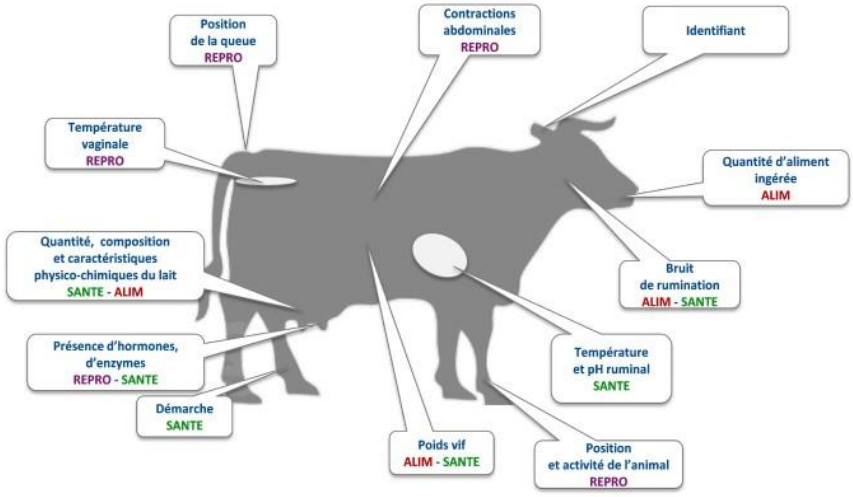


Figure 2 : Capteurs embarqués sur l'animal : leur localisation, les variables mesurées et les fonctionnalités attendues

SOURCE : C. ALLAIN, « Panorama des capteurs en élevage bovin », Institut de l'Élevage, 2012. [3] © idele-Droits réservés

QUESTIONS

En vous aidant de l'Annexe A :

1. Indiquez quels sont les types de capteurs utilisés pour mesurer les différents paramètres représentés en Figure 2.

2. Pour chaque capteur, expliquez quelle est la mesurande et la valeur réellement mesurée.

REMARQUE 1 (Différence entre mesurande et grandeur mesurée)

La mesurande est la grandeur que l'on veut mesurer. Cette dernière n'est pas toujours celle que l'on mesure ou que l'on peut mesurer.

Par exemple, dans le cas où la mesurande est la température d'une pièce (en ˚C), la grandeur mesurée est la tension aux bornes du capteur de température (en V).

2. TD : Système de numération

Dans ces différents exercices, vous serez amenés à utiliser les systèmes binaire et hexadécimal pour décoder des informations provenant de capteurs.

2.1. Pré-requis

Un système de numération est une convention utilisée pour compter. Dans un système de numération de base $b > 1$, les symboles $0, 1, 2, \ldots, b-1$ sont appelés chiffres. En utilisant ces chiffres, tout nombre entier positif N peut être représenté par une expression de la forme :

$$N = a_0 b^0 + a_1 b^1 + \ldots + a_n b^n$$

avec $n > 0$, et, pour $i = 0 \ldots n$, $a_i \in \{0, 1, 2, \ldots, b-1\}$ avec $a_n \neq 0$.

Pour noter le nombre N défini précédemment, en base b, la convention suivante est utilisée : $N = (a_n a_{n-1} \ldots a_1 a_0)_b$.

En utilisant cette dernière notation, en base 10, le nombre 2017 s'écrit :

$$(2017)_{10} = 2*10^3 + 0*10^2 + 1*10^1 + 7*10^0$$

En informatique/électronique, on utilise le système de numération binaire (base 2). Dans ce cas, les symboles utilisés sont 0 et 1. Les nombres écrits en base 2 sont appelés *binary digits* (digits binaires) ou *bits*. Pour reprendre l'exemple précédent :

$$(2017)_{10} = (11111100001)_2 = 1*2^{10} + 1*2^9 + 1*2^8 + 1*2^7 + 1*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 0*2^2 + 0*2^1 + 1*2^0$$

REMARQUE 2 (Bases supérieures à 10)
En informatique, la base 16 est souvent utilisée pour remplacer l'utilisation de nombres binaires, souvent constitués d'un nombre important de bits. Or, il n'existe que 10 chiffres (0, 1, 2, ..., 9). Par convention, les chiffres utilisés pour les bases supérieures à 10 sont empruntés à l'alphabet. Ainsi, la base 16 utilise les symboles $\{0, 1, 2, \ldots, 9, A, B, C, D, E, F\}$ comme chiffres.

REMARQUE 3 (Autres notations)
Souvent le préfixe 0*b* est utilisé devant un nombre pour désigner des nombres écrits en système binaire (base 2) et 0*x* en système hexadécimal (base 16). Sans préfixe, on considère que le nombre est écrit en base 10. Exemple : 10 = 0*b*1010 = 0*x*A.

Le tableau représenté en Figure 3 donne les conversions des 16 premiers nombres en base 2, 10 et 16. Pour plus d'informations sur les systèmes de numération, voir par exemple le complément de cours de Christian BULFONE [4].

Décimal	Binaire	Hexadécimal
0	0b0000	0x0
1	0b0001	0x1
2	0b0010	0x2
3	0b0011	0x3
4	0b0100	0x4
5	0b0101	0x5
6	0b0110	0x6
7	0b0111	0x7

Décimal	Binaire	Hexadécimal
8	0b1000	0x8
9	0b1001	0x9
10	0b1010	0xA
11	0b1011	0xB
12	0b1100	0xC
13	0b1101	0xD
14	0b1110	0xE
15	0b1111	0xF

Figure 3 : Tableau de conversion décimal-binaire-hexadécimal

2.2. Quelques conversions de système de numération

2.2.1. Conversion en système de numération décimal

Pour convertir des nombres en système décimal, il faut sommer les puissances de la base initiale correspondante. Par exemple : $(101)_2 = 1*2^2+0*2^1+1*2^0 = (5)_{10}$

QUESTIONS

Effectuez les conversions des nombres suivants en système décimal :

1. $(1010)_2$

2. $(ABBA)_{16}$

3. $0b10100111001$

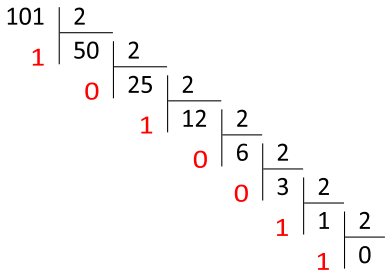
4. $(1982)_{16}$

5. $0xACDC$

2.2.2. Conversion en système de numération binaire ou hexadécimal

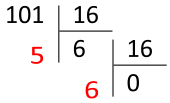
Pour convertir des nombres du système décimal vers le système binaire ou le système hexadécimal, il faut effectuer des divisions entières successives par la base considérée. La condition d'arrêt correspond à un quotient nul. Le nombre dans le nouveau système de numération est obtenu en lisant les restes du dernier vers le premier.

Par exemple, pour convertir $(101)_{10}$ dans le système de numération binaire, on effectue le calcul suivant :



Il en découle $(101)_{10} = (1100101)_2$.

De même, pour convertir le même chiffre en hexadécimal, on effectue le calcul suivant :



Il en découle $(101)_{10} = (65)_{16}$.

QUESTIONS

Effectuez les conversions des nombres suivants en système binaire et hexadécimal :

1. 1394

2. 2001

2.2.3. Conversion du système binaire en système hexadécimal

Pour convertir des nombres du système binaire vers le système hexadécimal, il faut remplacer, de droite à gauche, chaque groupe de 4 bits par le chiffre hexadécimal correspondant.

Par exemple, pour convertir $0b110000$ en système hexadécimal, on a :

$$\begin{array}{cc} 0b & \underbrace{0011} \quad \underbrace{0000} \\ 0x & 3 \quad\quad 0 \end{array}$$

Donc $(110000)_2 = (30)_{16}$.

2.2.4. Conversion du système hexadécimal en système binaire

Pour convertir des nombres du système hexadécimal vers le système binaire, il faut remplacer chaque chiffre hexadécimal par son équivalent binaire sur 4 bits (voir Figure 3).

Par exemple, pour convertir $0x14F$ en système binaire, on a :

0*x*	1	4	*F*
	0001	0100	1111

Donc $(14F)_{16} = (101001111)_2$.

QUESTIONS

Effectuez les conversions des nombres suivants en système binaire ou hexadécimal :

1. $(1394)_{16}$

2. $(1001)_2$

3. $(1111)_{16}$

4. $(101)_2$

2.3. Étude d'un capteur de température

Afin de réguler la température dans un bâtiment d'élevage de volailles, on utilise un capteur de température pour obtenir la température ambiante. On choisit d'utiliser le capteur de température de SK Pang [7]. Celui-ci utilise un bus de données[1] (un bus CAN) pour transmettre les informations de température provenant de deux sondes.

Avant de réguler la température, il faut être capable d'analyser l'information provenant du capteur. C'est l'objectif de cet exercice : à partir de l'information envoyée par le capteur, on souhaite obtenir la température.

La documentation technique indique que les informations des sondes de températures sont codées sur 17 bits chacune (voir Tableau 1) [7]. De plus, le constructeur indique que si le bit 0 de statut est 0 alors la donnée de température peut être utilisée, sinon il y a eu un problème de mesure.

Afin d'obtenir la valeur de la température, le constructeur indique qu'il faut multiplier le nombre codé en système binaire, par 0.25 une fois celui-ci converti en système décimal. Cela permet d'avoir une température à une résolution de $0.25\,^{\circ}\mathrm{C}$.

Donnée de température (16 bits)	Statut (8 bits)

Tableau 1 : Décomposition des 24 bits d'information du capteur de température

QUESTIONS

En analysant le message du capteur de température, on obtient les deux messages de température suivants :

Capteur 1 : $0x004F00$

Capteur 2 : $0x00CC01$

En vous aidant de la description précédente, répondez aux questions suivantes :

1. Convertissez les deux nombres en système binaire.

2. Pour chaque capteur, dites si la donnée de température est exploitable.

3. Donnez la valeur de la température de chaque capteur dont la donnée est exploitable.

a. Convertissez la donnée en système décimal.

b. Utilisez l'information du constructeur pour obtenir la valeur de la température mesurée.

1. Un système permettant à plusieurs périphériques de communiquer entre eux.

3. TP : Initiation au développement informatique pour l'acquisition de données

ATTENTION : Pour suivre ce TP, vous devez avoir quelques bases en programmation. Vous pouvez consulter, par exemple, le cours [6].

Objectifs et matériel utilisé :

L'objectif de ce TP est de vous initier au développement sur Arduino et à l'utilisation du langage de programmation Python pour acquérir des données. Pour l'exemple, un capteur de température sera utilisé.

Les objectifs de la séance sont :

1. Comprendre la programmation sur Arduino.
2. Mettre en place une solution d'acquisition de la température.
3. Utiliser Python pour tracer l'évolution de la température.

Afin de compléter ces objectifs, vous utiliserez le matériel suivant :

- Une carte Arduino UNO et son câble USB.
- Un capteur de température (AM2321 [5] ou équivalent).
- Deux résistances $4.7\ k\Omega$.
- Un ordinateur avec les logiciels Arduino et Python installés.

3.1. Le système d'acquisition

3.1.1. Premiers pas avec la carte Arduino

Le système d'acquisition utilisé par la suite est basé sur une carte Arduino. C'est une carte électronique comportant un microcontrôleur qui peut être programmé pour analyser et produire des signaux numériques et analogiques [10]. Ces signaux permettent, entre autres, d'effectuer de la domotique (le contrôle des appareils domestiques – éclairage, chauffage, ...), de la robotique (pilotage d'un robot) et, dans notre cas, de l'élevage de précision.

Il existe de nombreux types de cartes Arduino. Cette diversité permet d'avoir des cartes optimisées pour chaque application.

Dans ce module, nous avons choisi la carte Arduino UNO pour sa simplicité d'utilisation. Le Tableau 2 résume ses caractéristiques. Ce qu'il faut retenir, c'est qu'elle est basée sur un microcontrôleur 8 bits cadencé à 16 MHz : le microcontrôleur ATmega328P. La UNO dispose de 14 entrées/sorties digitales ainsi que 6 entrées analogiques [10].

La carte Arduino peut être alimentée via un ordinateur externe, un chargeur branché sur le secteur ou bien par le biais d'une batterie. Cette dernière méthode d'alimentation est utile pour effectuer du prototypage rapide. Lorsque la carte est reliée à un ordinateur, ce dernier a la possibilité de communiquer avec les fonctionnalités embarquées de la carte. Afin de programmer la carte, vous serez amenés à la brancher à un ordinateur.

QUESTIONS

En vous aidant de l'Annexe B :

1. Faites clignoter la LED, à une fréquence[2] de 1Hz, branchée sur la broche (*pin* en anglais) 13.

2. Faites afficher un message sur le moniteur série à chaque changement d'état de la LED (« ON », « OFF »).

REMARQUE 4 (Afficher un message dans le moniteur série)
Les méthodes Serial.println() et/ou Serial.print() permettent d'afficher un message dans la fenêtre du moniteur série (Outils->Moniteur série).

2. La fréquence est l'inverse de la période : une fréquence de 1Hz signifie que la LED doit clignoter 1 fois par seconde.

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Tableau 2 : Caractéristiques de l'Arduino UNO

3.1.2. Le capteur de température

Maintenant familiarisé avec la carte Arduino, on se propose de mettre en place un système d'acquisition de température, grâce au capteur de température.

Pour cela, vous allez d'abord câbler le système, puis vous écrirez le programme permettant de récupérer l'information provenant du capteur. Enfin, cette donnée de température sera envoyée vers l'ordinateur, grâce à la liaison USB.

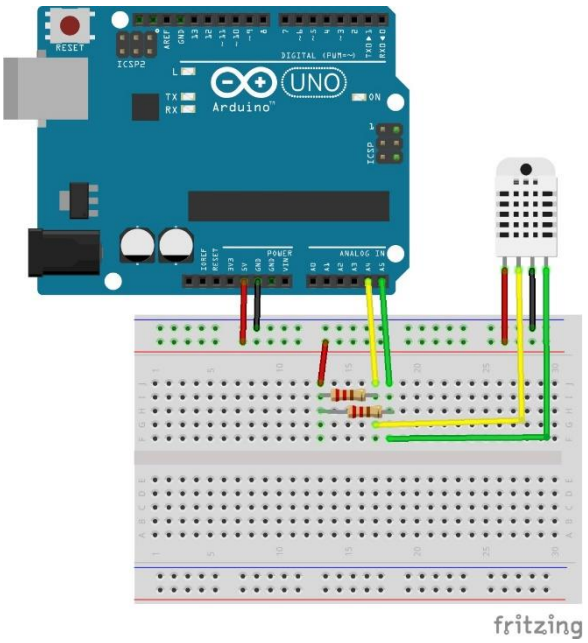


Figure 4 : Schéma de câblage du capteur de température

QUESTIONS

1. Observez le schéma de câblage représenté en Figure 4. En vous aidant de ressources documentaires, expliquez à quoi servent les deux câbles branchés sur les broches A4 (SDA) et A5 (SCL) de la carte Arduino.

2. Branchez les éléments comme présentés sur la Figure 4.

3. Écrivez le programme Arduino permettant de gérer l'information de température :

a. Créez un nouveau programme Arduino.

b. Sachant que la librairie de gestion du capteur de température AM2321 s'appelle AM2321.h, modifiez votre programme pour inclure cette librairie.

c. Sachant que pour utiliser le capteur, dans votre code, il faut déclarer et initialiser une variable en utilisant le formalisme « AM2321 am2321; », modifiez votre programme pour déclarer la variable permettant la gestion du capteur. Cette déclaration de variable devra être faite après la ligne d'importation de la librairie et avant la fonction « void setup(void) ».

d. Sachant que pour demander la lecture du capteur de température, il faut utiliser la méthode « am2321.read(); », modifiez votre programme pour envoyer un ordre de lecture du capteur, dans la procédure « void loop(void) ».

e. Sachant que la méthode de lecture du capteur renvoie un booléen (vrai –1– ou faux –0–), modifiez votre programme pour gérer le retour de l'ordre de lecture du capteur : si une acquisition a été faite, il faudra traiter l'information, sinon il ne faudra rien faire.

f. Sachant que l'information de température lue est accessible via la propriété « am2321.temperature » et correspond à 10 fois la valeur de température mesurée, modifiez votre programme pour traiter l'information du capteur et l'envoyer, vers l'ordinateur, par le port série.

4. Testez votre programme en le téléversant sur la carte Arduino et en ouvrant le moniteur série.

3.2. Le tracé des données

Bien que le logiciel Arduino permette de tracer des données provenant du port série, grâce à la fonctionnalité *Traceur série*, celle-ci souffre de quelques imperfections.

Dans cette partie, l'objectif est d'utiliser le langage de programmation Python pour :

1. Récupérer les informations du port série, en provenance d'une carte Arduino, pendant une période fixée.
2. Sauvegarder ces informations dans un fichier texte.
3. Tracer les informations obtenues.

Chaque point sera traité, dans l'ordre, dans les parties suivantes. On ne demandera pas de créer le script de toutes pièces. Une trame du script Python est donnée en Annexe C. Celle-ci comporte certaines erreurs. L'objectif est, pour chaque étape, de comprendre le script et de le corriger, si besoin.

3.2.1. Initialisation du script et des variables

Pour que le script Python s'exécute sans problème, il faut lui indiquer quels vont être les modules[3] utilisés par le script.

Dans notre exemple, il faudra, entre autres, utiliser un module permettant la gestion du port série.

QUESTIONS

Après une première lecture du script, répondez aux questions suivantes :

1. Quelles sont les lignes définissant les modules utilisés et les variables générales du script ?

2. Quels sont les modules importés dans le script ? En utilisant la documentation en ligne de Python, indiquez quelles sont leurs utilités.

3. Après avoir lu les objectifs du script présentés précédemment, expliquez quels sont les rôles des variables serialPort, tempsAcquisition et nomFichier.

3. Un module (ou librairie) Python permet, comme avec le programme Arduino, d'utiliser des fonctionnalités supplémentaires.

3.2.2. Python et le port série

La ligne 25 du script permet de lire les informations qui arrivent par le port série. Cependant, pour que tout fonctionne correctement, il y a un certain nombre de lignes supplémentaires. On se propose ici de comprendre comment sont acquises les données du port série.

QUESTIONS

1. Quelles sont les lignes permettant de gérer les données provenant du port série ?

2. La ligne 19 permet de se connecter au port série nommé serialPort de l'ordinateur. Le nombre 9600 permet de gérer le baudrate[4]. En fonction de la configuration de la carte Arduino (et du programme que vous avez créé dans la partie précédente), modifiez ce paramètre.

3. Expliquez quels sont les rôles des variables t0, list_data et list_temps.

4. À quoi sert la ligne 24 ?

3.2.3. L'utilisation de fichier texte

Une fois les données acquises, il faut être capable de les sauvegarder dans un fichier pour les exploiter plus tard, si besoin.

QUESTIONS

1. Quelles sont les lignes permettant de gérer l'enregistrement des données ?

2. Après analyse du code, donnez les 3 étapes nécessaires à l'écriture dans un fichier, une fois les données mises en forme.

3. Après une première exécution du script et après l'analyse du fichier de sortie, expliquez le rôle des lignes 35 à 38.

4. Le baudrate définit la vitesse de transmission des données sur le port série.

3.2.4. La création de figure avec Python

Une fois les données acquises, elles peuvent être tracées sous la forme d'une courbe d'évolution de la température en fonction du temps.

La ligne 51 du script donné en Annexe C comporte la mention « *ligne à modifier* ». On se propose ici de modifier cette ligne pour permettre de tracer l'évolution temporelle de la température.

QUESTIONS

La syntaxe à utiliser pour tracer le graphique est de la forme : plt.plot(*x*,*y*,label="Légende")

1. Quelle est la donnée à utiliser en abscisse ? Par quoi devez-vous remplacer *x* ?

2. Quelle est la donnée à utiliser en ordonnée ? Par quoi devez-vous remplacer *y* ?

3. En fonction des réponses précédentes, modifiez la ligne 51.
Pensez à modifier la chaine de caractère Légende par un terme plus adapté à la courbe.

ANNEXES

A - Capteurs en élevage de précision

SOURCE : idele.fr [3] et Chambre d'Agriculture Bretagne [9]

Capteur	Utilisation	Constructeur
Accéléromètre	• Détection des chaleurs (augmentation de l'activité) • Surveillances des vêlages (soulèvement de la queue) • Détection des troubles de la santé (diminution de l'activité)	• Pedometer Afitag (Afimilk) • Crystal act (Fullwood Packo) • Lactivator (Nedap) • Rescounter 2 (GEA) • Heat Seeker 2 (Boumatic) • HeatPhone (Medria) • Heatime (Milkline/Créavia) • Qwes H (Lely) • Moo monitor (Dairy Master) • Heat Box (Gènes Diffusion) • Activité-mètre (DeLaval) • HeatWatch 2 (CowChips) • Alert'Vel (ALB Innovation) • Smart'Vel (RF Track/Créavia)
Vidéo surveillance	• Détection des chaleurs • Surveillances des vêlages	• Tourelle évolutive (DetecVel) • Optima (Camerail) • Sofie (Visionaute)
Capteur de température	• Surveillances des vêlages • Détection des troubles de la santé (température du rumen)	• Vel'Phone® (Medria) • Vel Box (Gènes Diffusion) • San'Phone (thermobolus de Medria) • Bolus SmaXtex® (SmaXtec animal care) – avec capteur de pH
Capteur de pression	• Surveillances des vêlages (contraction abdominale) • Détection des troubles de la santé (boiteries)	• Agrimonitor (Databel) • Step Metrix (Boumatic)
Microphones (temps de rumination)	• Surveillances des vêlages • Détection des troubles de la santé	• HR tag (SCR) • Heatime Ruminact (Milkline) • Qwes HR (Lely)
Cellules somatiques	• Détection des troubles de la santé (troubles infectieux et mammites)	• MQC-C (Lely) • Compteur OCC ou DCC (DeLaval)
Conductivité	• Détection des troubles de la santé (troubles infectieux et mammites)	• Madetec

B - Utilisation d'un Arduino

B.1 - L'environnement de programmation

« *Le logiciel de programmation des modules Arduino est une application Java, libre et multi-plateforme, servant d'éditeur de code et de compilateur, et qui peut transférer le firmware et le programme au travers de la liaison série (RS-232, Bluetooth ou USB selon le module). Il est également possible de se passer de l'interface Arduino, et de compiler et uploader les programmes via l'interface en ligne de commande.* »

(Wikipédia)

Le leitmotiv de la programmation sur Arduino est le développement collaboratif. Ainsi, chaque personne est capable de publier des librairies afin d'effectuer certaines actions. La programmation orientée objet est la base de la programmation sur Arduino. Pour effectuer une tâche, on regarde si un utilisateur a déjà créé une librairie et on l'utilise.

Le logiciel couramment utilisé pour programmer sur Arduino est le logiciel éponyme. De nombreuses fonctionnalités sont utilisables avec cet environnement de développement, vous pourrez les découvrir au fur et à mesure du TP. Il permet, entre autres, de réarranger le code pour le rendre plus lisible. Il fournit aussi de nombreux exemples pour apprendre à utiliser les librairies. Plus d'informations sont fournies sur le site web https://www.arduino.cc.

Le langage de programmation utilisé est le C++, légèrement modifié pour que l'utilisation des composants de l'Arduino en soit simplifiée.

ATTENTION ! En C/C++, il faut déclarer les variables à utiliser. En d'autres mots, avant d'utiliser une variable entière, une chaine de caractères ou n'importe quelle autre variable, il faut indiquer au programme que cette variable aura un certain type :

```
1 int monEntier ;
2 monEntier = 42 ;
```

Il existe plusieurs types de variables, en voici une liste succincte :

boolean C'est une variable à deux valeurs possibles : true ou false. Elle est utile pour les tests conditionnels.

int C'est une variable contenant un nombre entier.

float C'est une variable contenant un nombre décimal.

char C'est une variable prenant 1 byte (généralement 8 bits) de mémoire ; elle est utilisée pour stocker un caractère.

string C'est une variable contenant un tableau de char ; elle est utile pour stocker des mots.

Une plus grande description des fonctionnalités du C++, appliquées à l'Arduino, est disponible sur le site web : https://www.arduino.cc.

Le tableau 3 ci-après résume les différences entre certaines syntaxes de Python et du C/C++.

	Python	C/C++

Le commentaire	`# Je suis un commentaire en ligne`
	
	`"""`
	`Je suis un bloc de commentaire`
	`"""`
		`// Je suis un commentaire en ligne`
		
		`/*`
		`Je suis un bloc de commentaire`
		`*/`

La création d'un tableau	`monTableau = [1,2,3,4]`	`int monTableau[]= {1,2,3,4}`

La conditionnelle `if`	`if <condition>:`
	`  <instructions...>`
		`if (<condition>)`
		`{`
		`  <instructions...>`
		`}`

La boucle `for`	`for i in range(<debut>,<fin>[,<pas>=1]):`
	`  <instructions...>`
		`for (i=<debut>;i<<fin>;i=i+<pas>)`
		`{`
		`  <instructions...>`
		`}`
		`// Pour un <pas>=1:`
		`for (i=<debut>;i<<fin>;i++)`
		`{`
		`  <instructions...>`
		`}`

La boucle `while`	`while <condition>:`
	`  <instructions...>`
		`while (<condition>)`
		`{`
		`  <instructions...>`
		`}`

La déclaration d'une fonction	`def <nomFonction>(<paramètres>) :`
	`  <instructions...>`
		`<type de sortie> <nomFonction>(<paramètres>)`
		`{`
		`  <instructions...>`
		`}`

Tableau 3 : Comparaison entre des syntaxes courantes en Python et C/C++

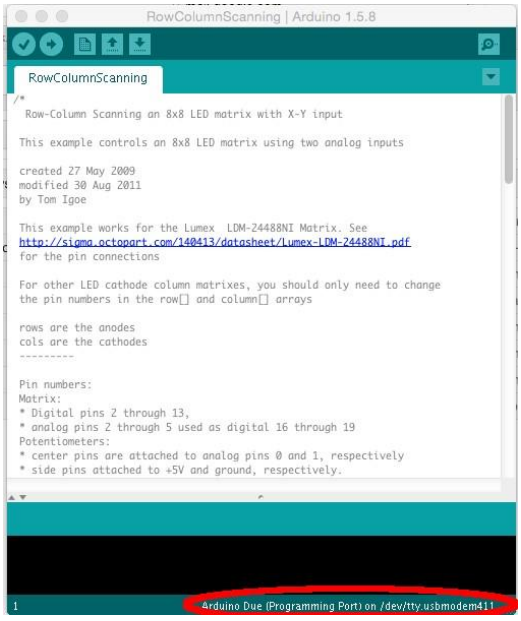


Figure 5 : Interface Arduino - Le port et la carte sélectionnés

B.2 - Quelques concepts de programmation temps réel (pour Arduino)

B.2.1 - L'environnement de développement

Pour commencer à programmer sur une plateforme temps-réel, il faut avoir une manière de communiquer avec celle-ci. Dans notre cas, ce sera avec l'application Arduino.

Après avoir connecté la carte Arduino sur un port USB, il faut ouvrir le programme Arduino.

Une fois ouvert, il faut sélectionner le port et le type de carte avec laquelle nous allons travailler (si ce n'est pas déjà fait). Le port et la carte que le programme utilise sont affichés en bas à droite de la fenêtre (voir Figure 5).

Il y a deux boutons importants sur l'IDE (interface de programmation) : le bouton de vérification et celui de téléversement. Le premier est utile pour tester si le code contient des erreurs. Le deuxième téléverse le programme sur l'Arduino (il fait une vérification du code avant). La Figure 6 met en avant ces deux boutons.

Pour de plus amples informations sur l'environnement de développement, voir le site web Arduino : http://arduino.cc/en/Guide/Environment.

B.2.2. Les différents fichiers d'un projet Arduino

Il existe trois types principaux de fichiers utilisables dans un projet Arduino. Ces fichiers sont définis par leurs extensions :

• Le fichier .ino : C'est le fichier principal contenant, entre autres, la fonction d'initialisation et la procédure loop contenant le programme à exécuter indéfiniment.

• Les fichiers .h : Ce sont les fichiers « en-têtes » (« *headers* » en anglais). Ils contiennent uniquement la description de différentes fonctions : ce sont les *prototypes* des fonctions[5]. Ils indiquent :
– le type de valeur renvoyé par la fonction ;
– le nom de la fonction ;
– les types d'arguments.
La syntaxe générale d'un prototype de fonction est :
Type_de_donnee_renvoyee Nom_De_La_Fonction(type_argument1, type_argument2, ...);

• Les fichiers .cpp : Ces fichiers (parfois .c ou .cc) contiennent les définitions (l'implémentation) des différentes fonctions et méthodes définies dans les fichiers d'en-tête.
La déclaration de la fonction est effectuée via la syntaxe suivante :
type_de_donnee Nom_De_La_Fonction(type1 argument1, type2 argument2, ...){ liste d'instructions}
Le type_de_donnee représente le type de valeur que la fonction retourne (char, int, float...). Si la fonction ne renvoie aucune valeur, elle est précédée du mot-clé void.

Remarque 1 *Le prototype est une instruction, il est donc suivi d'un point-virgule.*
Remarque 2 (Appel de fonction) *Pour exécuter une fonction, il suffit de faire appel à elle en écrivant son nom suivi d'une parenthèse ouverte (éventuellement des arguments) puis d'une parenthèse fermée. Pour plus de facilité, on peut copier/coller le prototype déclaré dans le fichier d'en-tête et modifier le nom des arguments, par exemple :*

```
#include "lib_TP1.h" // Fichier contenant les prototypes utilisés
void setup(void) {
initialisation_Servo() ; // Premier appel de fonction, sans argument
}
void loop(void) {
bougeServo(POS_RIGHT) ; // Second appel de fonction, avec argument delay(1000); bougeServo(POS_CLOSE) ;
}
```

5. Le prototype d'une fonction est une description d'une fonction qui est définie plus loin dans le programme.

B.2.3. Initialisation et boucle infinie

Un programme temps réel est voué à être exécuté en continu, sans jamais s'arrêter. Pour gérer cela, l'architecture d'un code Arduino est divisé en deux procédures : setup() et loop().

Dans la première procédure, setup(), il faut initialiser toutes les fonctionnalités utilisées par la suite. C'est le cas, par exemple, des broches (ou pins) numériques ou de la communication sur port série. Les instructions de cette procédure sont exécutées une seule fois, au démarrage de la carte.

La deuxième procédure, loop(), contient les instructions qui seront exécutées indéfiniment, jusqu'au débranchement de la carte.

Voici un exemple de code qui fait clignoter une LED, branchée sur la pin 53.

```
int ledPin = 53; // La LED est branchée sur la pin 53

  // Procédure exécutée une seule fois au démarrage
  void setup()
  {
        pinMode(ledPin, OUTPUT); // On déclare la pin 53 comme une sortie numérique
        Serial.begin(9600); // initialisation de la communication série à 9600 bps
  }
// Procédure contenant le programme qui boucle indéfiniment
void loop()
{
        digitalWrite(ledPin, LOW); // On éteint la LED
        Serial.println("Nuit !"); // On affiche un texte dans la console

        delay(500); // On attends 500 ms
        digitalWrite(ledPin, HIGH); // On allume la LED
        Serial.println("Jour !"); // On affiche un texte dans la console
}
```

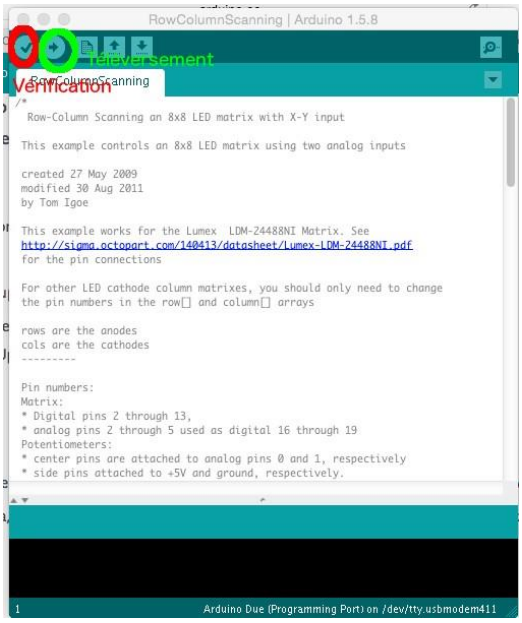


Figure 6 : Interface Arduino - Les boutons de vérification et de téléversement

B.2.4. Input/Output

Sur l'Arduino, il existe des broches d'entrée/sortie (INPUT/OUTPUT) par lesquelles nous pouvons transmettre des signaux numériques (avec deux états : HIGH et LOW) ou des signaux analogiques (qui varient entre 0 et 5V).

L'exemple de la section précédente montre comment utiliser une broche en sortie numérique avec les deux états : HIGH et LOW.

Voici un exemple d'utilisation d'une broche, en entrée analogique (la pin A0), qui fait allumer une LED si sa valeur dépasse un seuil.

```
int analogInPin = A0; // On regarde la pin analogique A0
int ledPin = 53; // La LED est branchée sur la pin 53

// Procédure exécutée une seule fois au démarrage
void setup()
{
        pinMode(ledPin, OUTPUT); // On déclare la pin 53 comme une sortie numérique

        // On définit la résolution des entrées analogique à 10 bits (0->1023)
        analogReadResolution(10);

        Serial.begin(9600); // initialisation de la communication série à 9600 bps
}


// Procédure qui boucle indéfiniment
void loop()
{
        if (analogRead(analogInPin)<512) // On lit l'entrée analogique
        {
                // Si l'entrée est inférieur à 512
                digitalWrite(ledPin, LOW); // On éteint la LED
                Serial.println("Nuit !"); // On affiche un texte dans la console

        }
        else
        {
                // Sinon
                digitalWrite(ledPin, HIGH); // On allume la LED
                Serial.println("Jour !"); // On affiche un texte dans la console
        }
}
```

B.2.5. Interruptions

Étant donné que les plateformes temps-réel sont, généralement, mono-tâche (on ne peut pas traiter plusieurs instructions en même temps), on utilise des interruptions pour gérer les priorités des processus.

Une interruption est une méthode mise en place sur le microcontrôleur qui permet d'exécuter une procédure lorsqu'un certain type de signal est émis. Ce signal peut être reçu sur une pin ou être virtuellement émis par une procédure (e.g. un timer qui serait arrivé au terme de son décompte).

Il existe plusieurs sortes d'interruption sur l'Arduino. Vous pouvez vous instruire sur le site web Arduino : http://arduino.cc/en/Reference/AttachInterrupt. Voici un exemple d'utilisation d'une interruption (sur la pin 2) qui fait allumer une LED lors de l'appui sur un bouton.

```
int pinButton = 2; // Le bouton est branché sur la pin 2

int pinLed = 13 ; // La LED est branchée sur la pin 13

void setup(void) {
        pinMode(pinLed, OUTPUT); // On déclare la pin 13 comme une sortie
        pinMode(pinButton, INPUT_PULLUP); // On déclare la pin 2 comme une entrée

        // On définit l'interruption qui exécutera la fonction manageSelectButton lorsque la pin passera à l'état
            bas.
        attachInterrupt(digitalPinToInterrupt(pinButton), manageSelectButton, LOW);
}

void loop(void) {
        // Des calculs...
}

void manageSelectButton() {
        // Cette fonction est exécutée quand la pin 2 est LOW
        digitalWrite(pinLed, HIGH) ;
        delay(100);
        digitalWrite(pinLed, LOW) ;
}
```

C - Trame du fichier Python de gestion de la température

```
#encoding: utf-8

import serial
import time
import matplotlib.pyplot as plt

#####################################################
# Déclaration des variables
#####################################################

serialPort = "COM4"
tempsAcquisition = 10
nomFichier = "out.csv"

#####################################################
# Acquisition des données
#####################################################

ser = serial.Serial(serialPort, 9600)
t0 = time.time()
list_data = []
list_temps = []
t = 0
while t <= tempsAcquisition:
    data_fromSerial = ser.readline()
    t = time.time()-t0
    list_temps.append(t)
    list_data.append(int(data_fromSerial))

#####################################################
# Enregistrement des données
#####################################################

# Préparation des données
data_txt = "Temps;Temp."
data_txt += "\n"
for idx in range(len(list_temps)):
    data_txt += "{};{}\n".format(list_temps[idx], list_data[idx])
# Ouverture du fichier
fichier = open(nomFichier, 'w')
# Ecriture des données
fichier.write(data_txt)
# Fermeture du fichier
fichier.close()

#####################################################
# Trace les données
#####################################################

figure = plt.figure()
# ligne à modifier
plt.xlabel("Temps (s)")
plt.ylabel("Température (° C)")
plt.legend()
plt.show()
```

RÉFÉRENCES

[1] Adafruit, « Adafruit PN532 RFID/NFC Breakout and Shield », Consulté en 2017.
https://learn.adafruit. com/adafruit-pn532-rfid-nfc

[2] C. Allain, A. Chanvallon, P. Clement, R. Guatteo, N. Bareille, « Élevage de précision : périmètre, applications et perspectives en élevage bovin », in *Rencontres autour des Recherches sur les Ruminants (21e ed.)*, p. 3-10, Institut de l'Élevage - Inra, 2014 [Présentation orale].

[3] C. Allain, « Panorama des capteurs en élevage bovin », Institut de l'Élevage, 2012, Consulté en 2016. https://idele.fr/detail-article/panorama-des-capteurs-en-elevage-bovin [Online; accessed January-2016]

[4] C. Bulfone, « Les systèmes de numération en base 2, 8 et 16 », Complément de cours, GIPSA-LAB, Consulté en 2017.
http://www.gipsa-lab.grenoble-inp.fr/~christian.bulfone/MIASHS-L3/PDF/systemes-numeration.pdf

[5] Aosong Electronics, « Digital Temperature and Humidity Sensor – AM2321 Product Manual », Consulté en 2017.
http://akizukidenshi.com/download/ds/aosong/AM2321_e.pdf

[6] T. Maillot, *M2207 – Programmation Orientée Objet*, 1re année R&T, IUT de Dijon-Auxerre, 1re edition, 2014. Polycopié de cours (18 pages).

[7] SK Pang, 2 Channel CAN-Bus Thermocouple Interface K-Type, July 2016.

[8] D. Paret, *RFID en ultra et super hautes fréquences : UHF-SHF - Théorie et mise en œuvre*, Électronique. Dunod, 2008.

[9] Y. Ramonet, C. Bertin, « Utilisation d'accéléromètres pour évaluer l'activité physique des truies gestantes logées en groupes. Développement de la méthode et utilisation dans six élevages au DAC », Rapport, Chambres d'agriculture de Bretagne, 2015.
http://www.synagri.com/ca1/PJ.nsf/46b50bbadf2cf901c1256c2f0041b9a7/d095a1d58cfcaa65c1257e3700531556/$FILE/Activité_Truies_accéléromètre_CRAB2015.pdf

Voir aussi :

A. Dumont, B. Yernaux, *50 montages pédagogiques avec Arduino*, Guide pédagogique, « Approches », 2017.