Etude d'un microcontrôleur avec l'Arduino UNO

Arnaud Dupont, Vincent Herb, Jeanne Donnée, Jeanne Pister, Tom Dukatenzeiler

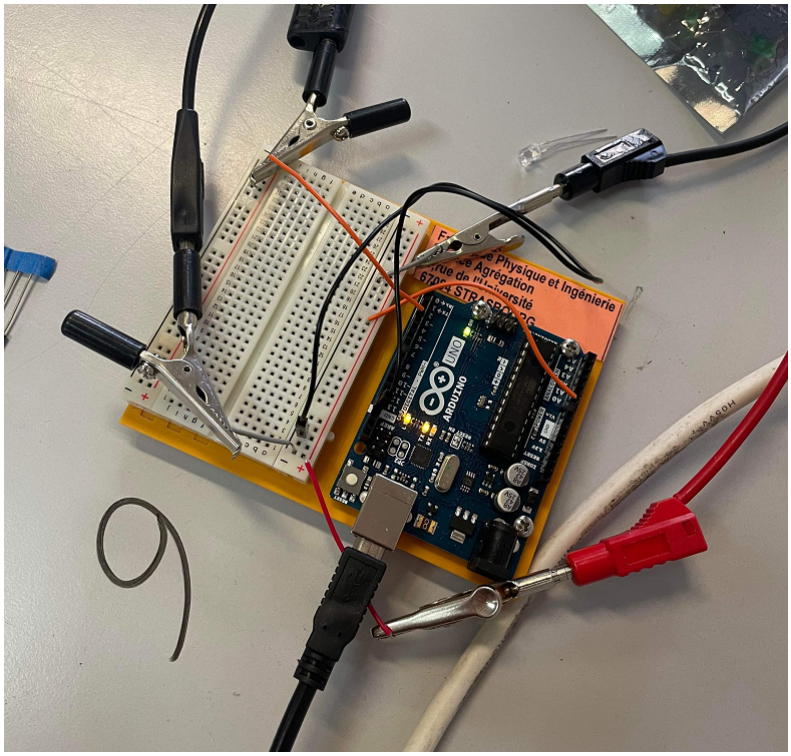
7 juillet 2022

Dans ce document, nous allons vous présenter quelques pistes permettant une étude générale des microcontrôleurs en se basant sur la carte Arduino UNO, un composant électronique de plus en plus utilisé non seulement par les bricoleurs du dimanche mais également par les professeurs et élèves de collèges et lycées. Les présentes manipulations ne s'adressent cependant pas forcément à des collégiens et lycéens malgré leur relative simplicité de mise en œuvre : elles peuvent cependant illustrer des notions abordées en classes préparatoires.

La carte Arduino UNO contient un certain nombre de composants : la pièce maîtresse de la carte est le microcontrôleur AT-Mega328P. Voir les spécifications constructeur pour plus de détails sur le site officiel d'Arduino.

Un microcontrôleur est un composant électronique, comportant entrées et sorties, et générant les signaux de sortie en fonction des signaux d'entrée et d'un script qui peut être stocké dans une mémoire.

L'étude des manipulations suivantes nécessite de déjà connaître l'architecture de la carte Arduino : repérez les entrées masse (GND), les entrées analogiques (A0, A1, etc.) les sorties numériques et les sorties numériques PWM. Vous devez également connaître certaines des instructions de base du langage Arduino : analogRead, digitalWrite, communication avec le moniteur série, etc.



Beaucoup des montages proposés auront une allure semblable à ceci.

1 La mémoire de l'Arduino UNO

Le microcontrôleur de l'Arduino UNO est doté d'une mémoire : il est donc possible d'enregistrer un script sur celle-ci et de modifier le comportement de la carte. Une fois téléversé depuis un ordinateur, la carte peut en être débranchée : elle conservera le script et fonctionnera de manière autonome en suivant les instructions fournies par l'utilisateur.

Une simple illustration de ceci consiste à faire clignoter la LED de l'Arduino selon des durées choisies par l'utilisateur. Cette expérience facile, proposée d'ordinaire à tous les débutants en Arduino met en évidence la mémoire de l'Arduino.

Cette expérience ne requiert aucun matériel excepté le câble USB permettant la connexion à un ordinateur, un ordinateur depuis lequel le script sera téléversé, et la carte Arduino UNO (la LED est déjà équipée sur la carte). On pourra, de manière facultative, y ajouter une photodiode et un oscilloscope. Le script est donné en annexe, il se trouve également inclus parmi les exemples fournis avec le logiciel Arduino.

Le délai séparant l'allumage ou l'extinction de la LED se règle avec les instructions `delay()` avec entre parenthèses un nombre indiquant le nombre de millisecondes qu'Arduino laisse passer avant d'exécuter l'instruction suivante. On pourra chercher à modifier les délais et à les mesurer à l'oscilloscope par exemple avec une photodiode.

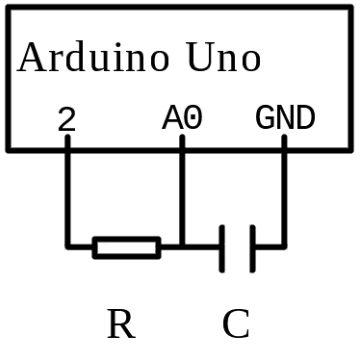
2 Application de l'Arduino : mesure de la constante de temps d'un circuit RC

Cette manipulation permet de montrer l'intérêt des microcontrôleurs. On utilise la mémoire des microcontrôleurs, leur capacité de gestion d'entrée et de sorties, et de mesure du temps (la carte Arduino est équipée d'un oscillateur à quartz servant d'horloge).

Matériel requis :

— Une carte Arduino UNO
— Un ordinateur
— Platine de prototypage et fils Dupont
— Plusieurs résistances de différentes valeurs (de 10^2 à $10^5\Omega$ environ)
— Plusieurs condensateurs de différentes valeurs (de 10^{-9} à $10^{-6}F$ environ)

Faire le branchement suivant :



Voir le script en annexe. Les branchements peuvent être adaptés à condition de modifier celui-ci convenablement.

Choisir de prime abord R et C de manière à avoir une constante de temps $\tau = R \times C$ de l'ordre du centième de seconde.

Téléverser le script dans la carte Arduino et allumer le moniteur série.

Le moniteur série montre plusieurs mesures successives de la constante de temps du circuit RC. On rappelle que lorsqu'un circuit RC voit sa tension d'alimentation passer brusquement de 0 à 5V, la tension aux bornes du condensateur est $5 \times (1 - \mathrm{e}^{-t/\tau})$ volts.

Pour $t = \tau$, la tension se trouve à $\approx 63,2\%$ de sa valeur finale.

Les étapes de l'exécution du script sont les suivantes :

— S'assurer que le circuit n'est plus alimenté (contrôle de la valeur lue sur A0).

— Lorsque le circuit n'est plus alimenté, lui faire subir un échelon de tension, via la sortie 2, en enregistrant l'instant t_0 où cet échelon se produit.

— Mesurer la tension aux bornes du condensateur via l'entrée A0. Lorsque cette tension atteint 63,2% de la valeur de l'échelon, on se trouve à $t = t_0 + \tau$: on enregistre l'instant où la tension atteint cette valeur.

— La valeur de τ est déduite des instants mesurés précédemment. L'alimentation du circuit RC est coupée et un nouveau cycle commence.

La carte effectue ainsi automatiquement des mesures, de manière autonome, selon un script qui lui est donné. On montre ainsi que la carte Arduino gère des sorties et des entrées, qu'elle a une mémoire, et qu'elle est dotée d'une horloge interne. Les mesures multiples de τ peuvent donner lieu à des incertitudes statistiques sur les valeurs des composants utilisés.

On pourra refaire plusieurs fois la manipulation en employant des valeurs de composants variées, pour voir les performances du dispositif lors de mesures de constantes de temps plus basses (millième de seconde, dix-millième de seconde, cent-millième de seconde...)

Vers les limitations des microcontrôleurs. On remarque que pour un produit $R \times C$ trop faible, l'Arduino n'est plus capable de donner une valeur fiable de τ. On met ainsi en évidence que le microcontrôleur présente certaines limites dans son utilisation. C'est l'objet des deux manipulations suivantes.

3 Le temps de réponse de l'Arduino UNO

On a pu voir que l'Arduino Uno avait des difficultés à gérer des temps trop courts. Il est également possible de mesurer son *temps de réponse.* En effet, lorsque l'Arduino reçoit un signal d'entrée, il peut prendre un certain temps avant de réagir. Ainsi, si on demande à Arduino d'allumer instantanément une sortie s'il reçoit un signal d'entrée, on aura tout-de-même un délai incompressible entre l'arrivée de l'entrée et l'allumage de la sortie.

Un script très simple permet de mettre en évidence cette propriété et de mesurer ce temps de réponse.

Matériel requis :

— Une carte Arduino UNO

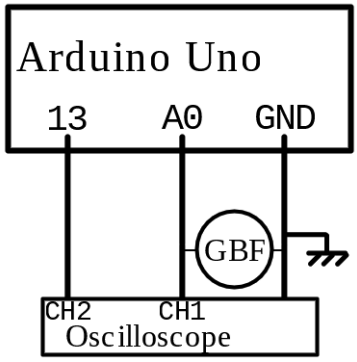
— Un oscilloscope

— Un générateur basses fréquences

— Connecteurs pour Arduino, platine de prototypage et fils Dupont

— Pinces crocodile et câbles coaxiaux

Faire le branchement suivant :

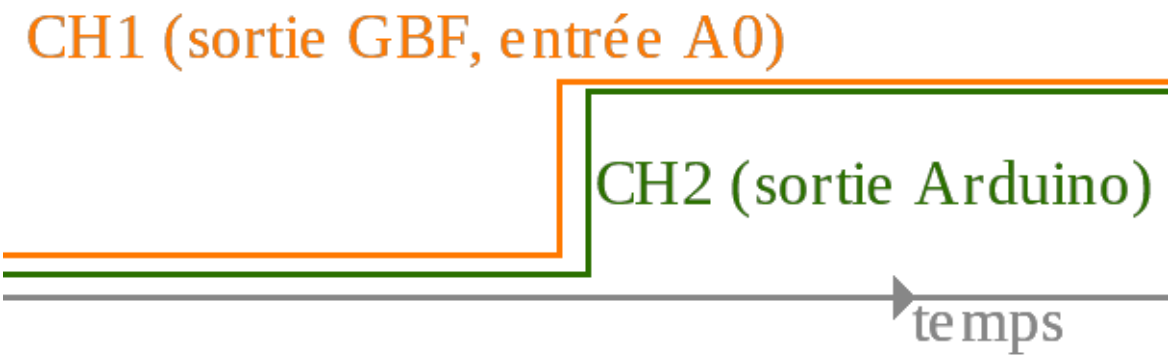


Le script se trouve en annexe.

Le GBF doit envoyer un signal créneau, oscillant entre 0V et une valeur supérieure (par exemple 4V ; évitez de dépasser les 5V dans tous les cas). On pourra régler la fréquence du GBF sur une vingtaine de Hertz pour commencer. Les étapes de déroulement du script sont les suivantes :

— Arduino lit la tension sur l'entrée A0.
— Si cette tension excède un certain seuil (500 sur 1023, correspondant à 2.44V), on allume la sortie 13.
— Dans le cas contraire, on éteint la sortie 13.

On observe à l'oscilloscope quelque-chose de ce style :



On observe un délai entre l'allumage de la sortie et l'arrivée de l'entrée. C'est le temps de réponse recherché. Plus intéressant : on constate que *celui-ci fluctue, de manière très régulière.* On peut changer la fréquence du signal du GBF et constater que la vitesse à laquelle fluctue le temps de réponse varie elle aussi. Ces fluctuations peuvent être imputées à l'échantillonage temporel sur les entrées analogiques de l'Arduino UNO. On trouve ainsi deux contributions au temps de réponse :

— Quand l'Arduino « comprend » qu'il a reçu un échelon de tension en A0, il met un certain temps avant de réagir et d'allumer la sortie. On pourrait qualifier ce temps de « temps de réponse intrinsèque ».
— En raison de l'échantillonnage, il est possible qu'entre l'arrivée réelle de l'échelon de tension sur A0 et sa détection s'écoule un certain délai. Si, par chance, l'échelon de tension arrive juste avant un échantillonnage sur A0, alors ce délai sera court. Cependant si l'échelon arrive immédiatement après l'échantillonnage sur A0, alors il faudra attendre jusqu'à l'échantillon suivant pour qu'Arduino détecte cet échelon. Ce délai est donc compris entre 0 secondes et la valeur du temps d'échantillonnage de l'Arduino.

Il est ainsi possible de réaliser deux estimations à partir de ce même dispositif. Tout-d'abord, réglez la fréquence du signal créneau sur le GBF de sorte que les fluctuations du temps de réponse soient assez lentes : faites pour cela varier par tâtons la fréquence sur des hertz ou des dixièmes de hertz, autour d'une valeur centrale de 20Hz par exemple.

Mesurez, à l'oscilloscope, le délai le plus court entre l'échelon de tension et l'allumage de la sortie Arduino. Ceci vous donne le temps de réponse intrinsèque, qui est, d'après la documentation Arduino, environ 10 microsecondes.

Mesurez ensuite la différence entre le délai le plus court et le délai le plus long : on obtient ainsi une estimation du temps d'échantillonnage de l'Arduino. Il est d'environ 100 microsecondes (la fréquence d'échantillonnage est de 10kHz).

Remarque. En vérité le temps de réponse, et peut-être le temps d'échantillonnage, dépendent du code enregistré dans la mémoire de l'Arduino. Une communication avec le moniteur série aura tendance à faire augmenter le temps de réponse de manière assez spectaculaire. Les temps

de réponse et d'échantillonnage ainsi mesurés ne sont donc pas forcément définitifs puisqu'ils dépendent de la situtation dans laquelle l'Arduino UNO fonctionne.

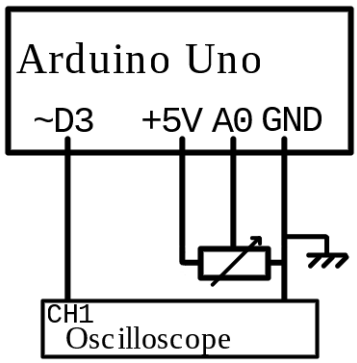
4 Autres quantifications dans l'Arduino UNO

Il est également possible de mettre en évidence une quantification sur le rapport cyclique des sorties PWM. Ces sorties délivrent un signal créneau, oscillant entre 0 et +5 volts, et dont le rapport cyclique peut prendre 256 valeurs, les extrêmes étant 0 (signal perpétuellement éteint) et 100% (signal perpétuellement à +5V). La manip suivante propose un moyen de retrouver le nombre de bits sur lequel est codée cette valeur.

Matériel requis :

— Une carte Arduino UNO
— Un potentiomètre (quelques kΩ)
— Une platine de prototypage et fils Dupont
— Un câble coaxial, des pinces crocodile
— Un oscilloscope

Faire le branchement suivant :



Le script est donné en annexe.

Le but de la manœuvre est d'imposer une tension contrôlable, analogique, sur l'entrée A0. On a dans cette configuration, une tension pouvant varier de 0 à +5V selon la position du potentiomètre.

L'Arduino va lire cette tension et s'en servir pour commander la sortie PWM. Le signal de sortie est visualisé à l'oscilloscope.

Petit problème : on cherche à visualiser la quantification uniquement sur la sortie PWM. Or, il y a déjà une quantification sur l'entrée A0. Pour « s'affranchir » de cette quantification, on divise la valeur de la tension lue sur A0 (entier entre 0 et 1023) par 10 (on obtient un nombre décimal avec 1 chiffre après la virgule, compris entre 0 et 102,3), de sorte que la quantification sur la valeur de la tension ne soit plus gênante (le pas de quantification devient suffisamment fin).

Téléversez le script, allumez le moniteur série et visualisez la sortie PWM à l'oscilloscope.

Faites varier la position du potentiomètre et observez les changements brusques du rapport cyclique sur le signal de sortie. Sur le moniteur série, on peut voir que ces changements interviennent à chaque fois que la valeur de commande (valeur donnée par A0 puis divisée par 10) voit sa partie entière changer.

Notez la période T du signal de sortie. Mesurez ensuite l'incrément de temps passé dans l'état haut lors d'un changement de rapport cyclique τ.

Le nombre de possibilités pour l'allure du signal est $\frac{T}{\tau}+1$ (le +1 permettant de tenir compte de la possibilité du signal complètement éteint). On s'attend à trouver une valeur proche de 256.

On en déduit le nombre de bits N sur lesquels le rapport cyclique est encodé :

$$2^N = 256$$

$$N \ln(2) = \ln(256)$$

$$N = \frac{\ln(256)}{\ln(2)} = 8$$

Dans le cas où on ne trouverait pas exactement 256, on arrondirait N à l'entier le plus proche.

Conclusion

Il a été possible de voir, au cours des précédentes manipulations, que le microcontrôleur de la carte Arduino UNO est un composant électronique très pratique pour gérer de manière autonome et intelligemment (plus ou moins suivant l'intelligence de l'auteur du script enregistré dessus) des circuits électriques. Evidemment ce composant comporte aussi certaines limitations qu'il faut être capable de quantifier pour en tirer le meilleur. On a pu à cette occasion étudier des propriétés générales des composants électriques (temps de réponse) et réinvestir des notions de numérisation du signal (quantifications).

Bibliographie

Site officiel d'Arduino : https ://www.arduino.cc/
Page dédiée plus spécifiquement à l'Arduino UNO :
https ://docs.arduino.cc/hardware/uno-rev3
Quelques infos sont trouvables à la référence suivante :
Bellier, J.-P., Bouloy, C., Guéant, D. (2020). *Expériences de physique : Electricité, électromagnétisme, électronique, transferts thermiques.* Dunod.
Le code proposé pour l'expérience 2 est inspiré d'une ressource mise à disposition par l'académie de la Guyane :
https ://physique-chimie.dis.ac-guyane.fr/Tle-Spe-PC-Theme-4-TP-Constante-de-temps-d-un-circuit-RC-python-arduino.html

Annexe 1

Code pour l'expérience 1 : LED clignotante

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Code inclus parmi les exemples d'introduction dans le logiciel Arduino (programme Blink)

Annexe 2

Code pour l'expérience 2 : mesure de la constante de temps d'un circuit RC

```
//Ce script comprend une boucle permettant de lancer plusieurs mesures
//de la constante de temps d'un même circuit à la suite.
//On alimentera le circuit au moyen de la broche notée pinU (voir variables plus bas) ;
//la tension aux bornes du condensateur sera lue au moyen de l'entrée A0.

//Déclaration des variables utilisées
unsigned long temps0, temps1;  // Deux variables de temps pour mesurer tau
float tau;  // Constante de temps
int pinU = 2;  // Broche alimentant le circuit RC


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600) ;  // Initialisation de la communication série
  pinMode(pinU, OUTPUT);  // On paramètre broche d'alimentation du circuit en Sortie
  digitalWrite(pinU, 0);  // On éteint la broche en question (tension nulle sur le circuit)
}


void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Début d'une nouvelle mesure"); //On prépare le lancement de la mesure.
  Serial.println("Contrôle de la décharge du condensateur");
  while (analogRead(A0) > 0) {  // Tant que le condensateur n'est pas complètement déchargé :
    Serial.println(analogRead(A0));  // Affiche la valeur lue à l'entrée reliée au condensateur.
    delay(500);
  }


  Serial.println("Alimentation du circuit"); //Quand le condensateur est complètement déchargé, on lance la mesure.
  delay(500);
  temps0 = micros();  // On relève t0
  digitalWrite(pinU, 1);  // On applique 5V en entrée du circuit RC.
  while (analogRead(A0) < 647) {  // Tant que le niveau lu est inférieur à 647 (63.2% de 1023, pour lequel t = tau)
    //On attend
  }

  temps1 = micros();  // Une fois sorti de la boucle on mesure t1

  tau = (temps1 - temps0) * 1e-6;  // Calcul de tau, converti en secondes
  Serial.print("Constante de temps du circuit en secondes : ");
  Serial.println(tau, 3); // On affiche tau avec 3 chiffres après la virgule.
  digitalWrite(pinU,0); //On éteint l'alimentation du circuit.
  delay(10000); //On attend avant de relancer une mesure.
}
```

Annexe 3

Code pour l'expérience 3 : mesure du temps de réponse de l'Arduino UNO

```
//L'objectif de ce script est de permettre la mesure du temps de réponse de l'Arduino face à une sollicitation
//(un échelon de tension arrivant sur une de ses entrées)
//Pour-ce-faire, on lui donne comme instructions d'allumer une sortie si la tension en entrée excède un seuil.
void setup() {
  // put your setup code here, to run once:
//Serial.begin(9600); //Cette ligne lance la communication avec le moniteur série (décommenter pour activer).
//Attention car cela influe sur le temps de réponse.
}

void loop() {
  // put your main code here, to run repeatedly:
int tensionlue = analogRead(A0); //Lit la tension sur l'entrée analogique A0
//Serial.println(tensionlue); //Cette ligne permet de lire en temps réel l'entrée A0
//sur le moniteur série (décommenter pour activer).
//Mais cela fait augmenter le temps de réponse

if (tensionlue > 500) { //Si cette tension est supérieure à un seuil
  digitalWrite(13, HIGH); //Allume la sortie 13
}
if (tensionlue <=500) { //Sinon
  digitalWrite(13, LOW); //Eteins la sortie 13
}
}
```

Annexe 4

Code pour l'expérience 4 : étude de la quantification des sorties PWM de l'Arduino UNO

```
//L'objectif de ce script est de mettre en évidence la quantification du rapport cyclique
//sur les sorties PWM de l'Arduino.
//Pour cela, on demande à Arduino de moduler le rapport cyclique d'une de ses sorties en fonction
//d'une tension qu'il lira en entrée, qu'on se chargera de faire varier de manière continue.

int sortiemodulee = 3; //On choisit la sortie PWM numéro 3 pour cette expérience

void setup() {
  // put your setup code here, to run once:
Serial.begin(9600); //Lance la communication avec le moniteur série
pinMode(sortiemodulee, OUTPUT); //Choisit le mode pour la sortie choisie
}

void loop() {
  // put your main code here, to run repeatedly:
float tensionlue = analogRead(A0); //Lit la tension sur l'entrée A0 (encodée sur un nombre entre 0 et 1023)
float commande = tensionlue/10;
//On divise ce nombre afin de s'affranchir de la quantification
//créée par la conversion analogique-numérique sur l'entrée A0.
Serial.println(commande); //Affiche cette tension sur le moniteur série
if (commande <200) { //Si la tension est inférieure à un seuil (cette condition n'est pas indispensable)
  analogWrite(sortiemodulee, commande); //Attribue la valeur de commande à la tension alternative
  //sur la sortie choisie (change le rapport cyclique)
}

}
```