

TP2 : circuit RC ; Arduino

Charge et décharge d'un condensateur en utilisant une carte Arduino**Document 1** Carte Arduino

Une carte Arduino est un microcontrôleur programmable, c'est-à-dire un petit ordinateur simplifié construit autour d'un circuit intégré unique contenant : un processeur, une mémoire et des entrées et des sorties programmables.

Cette carte permet de traiter des informations provenant de capteurs connectés à ses entrées, puis de commander des actionneurs reliés à ses sorties grâce à un programme.



Ainsi, une carte Arduino permet d'interagir avec le monde réel et de concevoir des systèmes automatisés ou des robots capables de réagir à leur environnement.

Une carte Arduino peut par exemple utiliser comme capteurs : des accéléromètres, des capteurs de température, des capteurs de pression atmosphérique, des capteurs de luminosité, etc.

Elle peut également commander différents actionneurs, tels que : des moteurs, des électroaimants, des DEL, des éléments chauffants, des haut-parleurs, des ventilateurs, des pompes, des vannes hydrauliques ou pneumatiques, etc.

Montage expérimental

Le montage présenté en ANNEXE permet d'étudier la **charge et la décharge d'un condensateur**.

Les bornes 8 et GND du microcontrôleur constituent le générateur du circuit RC. Elles fournissent une tension d'alimentation $E = 5,0\text{ V}$: la borne 8 correspond au pôle positif (+) et la borne GND au pôle négatif (-) du générateur.

Les bornes A0 et GND permettent de mesurer la tension aux bornes du condensateur.



Les condensateurs utilisés au lycée sont des condensateurs polarisés : leurs bornes positive (+) et négative (-) doivent être respectées

La borne négative (-) du condensateur doit être reliée à la borne GND du microcontrôleur.

Document 2 Entrées analogiques et convertisseur analogique-numérique d'Arduino

Les Arduino Uno possèdent des entrées analogiques, identifiées par les broches A0 à A5.

Les tensions appliquées sur ces broches sont toujours entre 0 et 5V. Ces tensions analogiques sont converties en valeurs numériques grâce au convertisseur analogique-numérique DAC (Digital Analog Converter) de la carte Arduino.

Ce convertisseur effectue une conversion sur 10 bits : la tension analogique mesurée sur une broche est transformée en un nombre entier compris entre 0 et 1023 (soit 2^{10} valeurs possibles)

- La valeur 0 correspond à une tension analogique 0 V
- La valeur 1023 correspond à une tension analogique de 5 V.

Dans les programmes Arduino utilisés pour l'acquisition des données, « Charge_condensateur.ino » et « Décharge_condensateur.ino », la fonction *analogRead(A0)* permet de lire la tension présente sur la broche

A0 et de la convertir en une valeur numérique comprise entre 0 et 1023, la carte Arduino traitant uniquement des données numériques.

1. Soit l'instruction $tension = analogRead(A0)$. Quelle est la tension analogique lue sur A0 (fraction de $E=5V$) en fonction de la valeur numérique $tension$?

Partie expérimentale

2. Dessiner le schéma du circuit électrique permettant d'étudier la charge et la décharge d'un condensateur.
3. Réaliser ensuite le schéma du montage utilisant la carte Arduino.

Charge du condensateur

- a) Réaliser le montage expérimental.
 - b) Ouvrir le programme « Charge_condensateur.ino ».
 - c) Téléverser le programme dans la carte Arduino.
 - d) Ouvrir le moniteur série (symbole de la loupe en haut à droite de la fenêtre Arduino).
 - e) Copier l'ensemble des valeurs obtenues dans le presse-papiers (CTRL + C).
 - f) Ouvrir le logiciel Regressi.
 - g) Créer un fichier à partir des données copiées en utilisant la commande : Fichier → Nouveau → Presse-papiers
4. À l'aide de Regressi, déterminer le modèle mathématique de la tension : $u_c(t)$. Retrouver la valeur de la constante de temps τ et en déduire la valeur de la résistance R .
 5. Mesurer la valeur de la résistance à l'aide d'un multimètre et comparer cette valeur à celle déterminée expérimentalement à la question précédente.
 6. Expliquer l'utilité de la ligne de programme “ $if(tension < 1019)$ ” (instruction conditionnelle).



Décharge du condensateur

Réaliser l'expérience à l'aide du programme « Decharge_condensateur.ino ».

Tracer la courbe de décharge dans Regressi, modéliser cette courbe et déterminer la valeur de la constante de temps τ .

Condensateur artisanal : déterminer l'épaisseur d'un film plastique

On place entre deux lames conductrices (en zinc ou en cuivre), reliées à un multimètre réglé en mode capacimètre, un film plastique alimentaire en polyéthylène dont on souhaite déterminer l'épaisseur.

Pour cela, on rapproche les deux lames conductrices afin de minimiser la distance entre elles, de manière à ce que les plaques soient au contact du film plastique. On appuie fermement sur les lames afin d'assurer un bon contact.

7. Mesurer la surface S des lames mises en vis-à-vis puis la capacité C du condensateur ainsi formé (en nF).

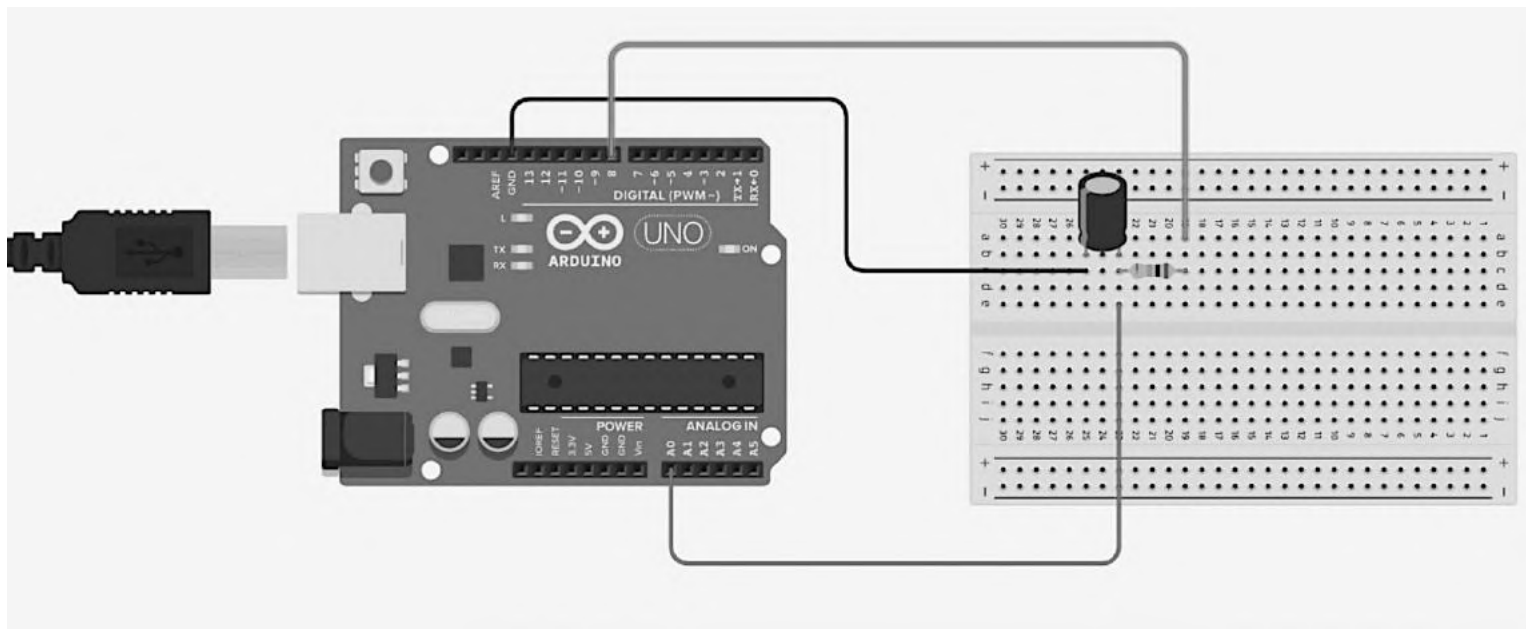
Données :

- Permittivité électrique du vide $\epsilon_0 = 8,85 \times 10^{-12} \text{ F.m}^{-1}$
 - Permittivité électrique relative de polyéthylène $\epsilon_r = 2,25$
 - Permittivité électrique d'un matériau $\epsilon = \epsilon_r \times \epsilon_0$.
8. Rappeler l'expression de la capacité C d'un condensateur plan en fonction de la surface S des plaques, de la distance e entre les plaques et de la permittivité ϵ du milieu isolant.
 9. À partir de cette relation, exprimer l'épaisseur e du film alimentaire en fonction de C , S et ϵ .



10. En supposant que le film est constitué de plusieurs couches de polyéthylène d'épaisseur moyenne de 3×10^{-10} m, estimer le nombre de couches composant l'épaisseur totale du film.

ANNEXE



Programme Arduino de charge d'un condensateur

```

long previousMillis = 0;
long interval = 10;
unsigned long topdepart = 0;
float tension;

// durée de millisecondes entre chaque mesure
// on initialise un variable topdepart à 0 ;
// variable tension qui sera lue

void setup()
{
  pinMode(8, OUTPUT);
  Serial.begin(9600);
  tension = analogRead(A0);

  // alimentation du condensateur
  // vitesse de communication avec arduino
  // on lit la tension sur la sortie analogique A0

  if (tension > 3) {
    Serial.println("décharge préalable du condensateur... patienter");
    while (tension > 3) {
      digitalWrite(8, LOW);
      delay(500);
      tension = analogRead(A0);
    }
    // on lit la tension sur la sortie analogique A0 toutes les 500 ms
  }
  Serial.println();
  Serial.println("*****");
  Serial.println();
  Serial.println("condensateur déchargé");
  Serial.println("La charge va démarrer, l'arrêt sera automatique quand le condensateur sera chargé");
  Serial.println("*****");
  delay(2000);
  topdepart = millis();
  digitalWrite(8, HIGH); // on met en marche l'alimentation
  // le top départ est initialisé sur la durée millis() du timer interne de l'arduino
  // on alimente le circuit pour charger le condensateur
}

void loop()
{
  unsigned long currentMillis = millis();

  // on crée une variable currentMillis qui correspond à millis() dans cette boucle

  tension = analogRead(A0);
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    // lorsque la durée courante - la durée précédente est supérieure à 10 ms (interval)
    // l'instant previousMillis devient CurrentMillis (réinitialisation du timer)

    if (tension < 1019) {
      float tension_lue = tension * 5.0 / 1023.0;
      //Serial.print("\t");
      Serial.print(millis() - topdepart);
      Serial.print("\t");
      Serial.println(tension_lue);
      // si cette tension est inférieure à 1018 sur 1023 (non complètement chargé)
      // tension analogique calculée
      // on sépare la donnée avec une tabulation (permettra de faire un csv par copier coller)
      // on écrit la durée millis-topdepart (première valeur à 0)
      // on sépare la donnée avec une tabulation (permettra de faire un csv par copier coller)
    }
  }
}

```

Programme Arduino de décharge d'un condensateur

```

long previousMillis = 0;
long interval = 10;
unsigned long topdepart = 0;
float tension;

// durée de millisecondes entre chaque mesure
// on initialise un variable topdepart à 0 ;
// variable tension qui sera lue

void setup()
{
  pinMode(8, OUTPUT);
  Serial.begin(9600);
  tension = analogRead(A0);

  // alimentation du condensateur
  // vitesse de communication avec arduino
  // on lit la tension sur la sortie analogique A0

  if (tension < 1019) {
    Serial.println("charge préalable du condensateur... patienter");
    while (tension < 1019) {
      digitalWrite(8, HIGH);
      delay(500);
      tension = analogRead(A0);
    }
    // on lit la tension sur la sortie analogique A0 toutes les 500 ms
  }
  Serial.println();
  Serial.println("*****");
  Serial.println();
  Serial.println("condensateur chargé");
  Serial.println("La décharge va démarrer, l'arrêt sera automatique quand le condensateur sera entièrement");
  Serial.println("*****");
  delay(2000);
  topdepart = millis();
  digitalWrite(8, LOW);
  // le top départ est initialisé sur la durée millis() du timer interne de l'arduino
  // on coupe l'alimentation du circuit pour décharger le condensateur
}

void loop()
{
  unsigned long currentMillis = millis();

  // on crée une variable currentMillis qui correspond à millis() dans cette boucle

  tension = analogRead(A0);
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    // lorsque la durée courante - la durée précédente est supérieure à 10 ms (interval)
    // l'instant previousMillis devient CurrentMillis (réinitialisation du timer)

    if (tension > 3) {
      float tension_lue = tension * 5.0 / 1023.0;
      //Serial.print("\t");
      Serial.print(millis() - topdepart);
      Serial.print("\t");
      Serial.println(tension_lue);
      // si cette tension est inférieure à 1018 sur 1023 (non complètement chargé)
      // tension analogique calculée
      // on sépare la donnée avec une tabulation (permettra de faire un csv par copier coller)
      // on écrit la durée millis-topdepart (première valeur à 0)
      // on sépare la donnée avec une tabulation (permettra de faire un csv par copier coller)
    }
  }
}

```