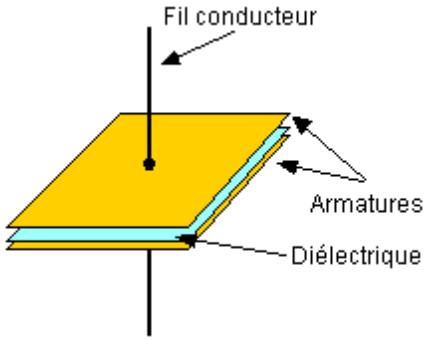
Condensateurs et dipôles RC

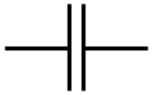
. Les condensateurs

Un condensateur est un composant électronique capable de stocker de l'énergie sous la forme d'un champ électrostatique. Il est constitué de 2 armatures séparées par un isolant, le "diélectrique".

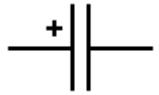


Les dimensions des armatures (ou plaques) sont grandes devant l'épaisseur de l'isolant. Par commodité, les films métalliques et l'isolant sont enroulés ou juxtaposés en plusieurs couches.

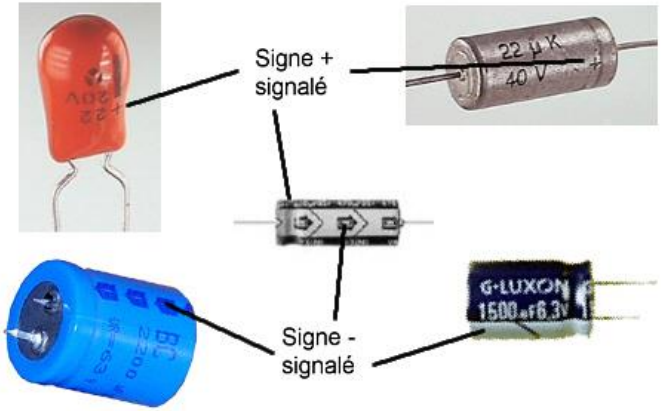
Dans les schémas de circuits électriques, le condensateur est symbolisé par :



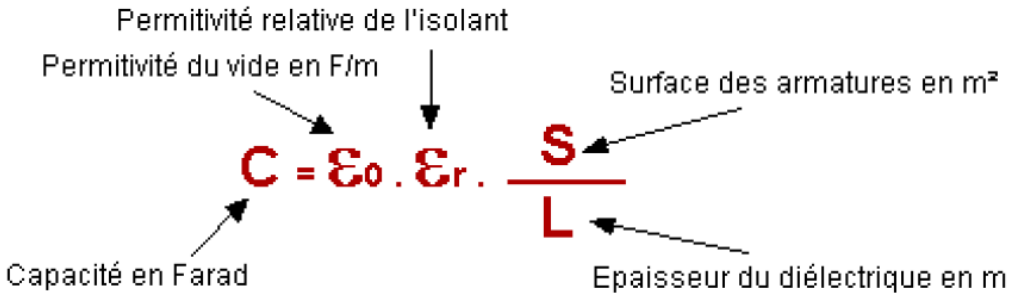
Les condensateurs avec un diélectrique chimique sont polarisés. Ils sont représentés par ce symbole :



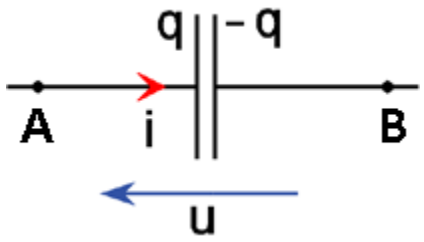
Les condensateurs polarisés possèdent un pôle "plus" et un pôle "moins", ils doivent impérativement être connectés dans le bon sens. En règle générale, les condensateurs polarisés radiaux (qui ont les deux pattes du même côté) possèdent une bande ou un ensemble de flèches qui désigne le pôle négatif, et les condensateurs polarisés axiaux (qui ont les deux pattes opposées) possèdent un renfoncement (collerette) côté pôle positif :



La caractéristique principale d'un condensateur est sa capacité exprimée en **farad.**



En appliquant, une tension U aux bornes d'un condensateur, il apparait une charge q, sur une de ses plaques, proportionnelle à la tension appliquée.



La charge qui apparaît sur l'une des plaques est l'opposé de celle qui apparaît sur l'autre plaque

En effet, la conservation de la charge électrique s'écrit : **q + q' = 0** d'où : **q' = – q**.

La charge q est, par convention la charge portée par la plaque par laquelle on entre dans le condensateur. On a alors :

q = C.U

où : q est exprimé en coulomb (C), U en volt (V) et C en Farad (F)

Remarque : Le farad étant une grande unité, on utilise souvent des sous-multiples :

$1\ \mu F = 10^{-6}\ F$, $1\ nF = 10^{-9}\ F$, $1\ pF = 10^{-12}\ F$

L'intensité du courant, i, est le débit de charges et est égale à la quantité de charges qui passent par unité de temps à travers une section de conducteur :

$$i = \frac{dq}{dt}$$

On oriente géométriquement le condensateur de la borne A vers la borne B.

Soit i_{AB} l'intensité algébrique du courant allant de A vers B et qA la charge qui apparaît sur la plaque par laquelle on entre dans le condensateur.

On a donc : **qA = – qB = C.(VA – VB)**

En tenant compte de la relation entre i_{AB} et qA, on a :

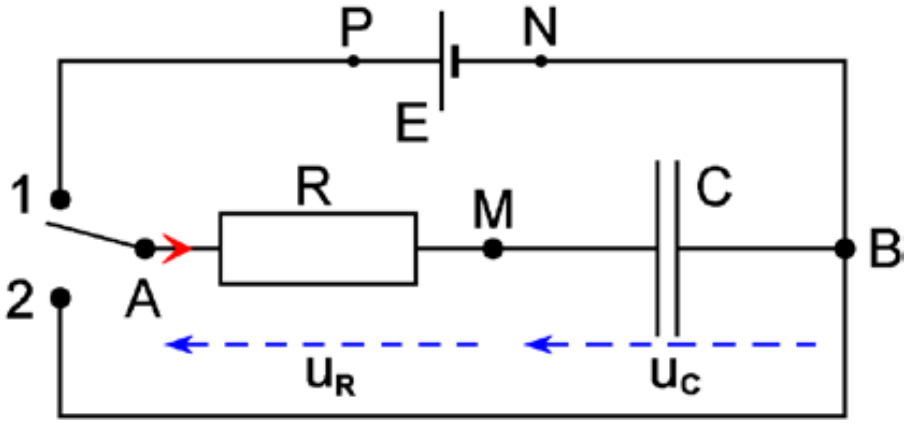
$$i_{AB} = \frac{dq_A}{dt} = C.\frac{d(V_A - V_B)}{dt} = C.\frac{du_{AB}}{dt}$$

On retiendra les relations suivantes :

$$u(t) = \frac{1}{C}.q(t) \quad \text{et} \quad i(t) = \frac{dq(t)}{dt} = C.\frac{du(t)}{dt}$$

. Le dipôle RC

Un dipôle RC est l'association en série d'un conducteur ohmique (conducteur ohmique) de résistance R et d'un condensateur de capacité C, comme dans le schéma ci-dessous :



1. Charge du condensateur

Le condensateur étant déchargé, on bascule, à l'instant t=0, l'interrupteur en position 1.

A chaque instant t > 0, on a :

$$\mathbf{V_A - V_B = u_R(t) + u_C(t) = R.i(t) + u_C(t) = E}$$

On a $i(t) = \frac{dq(t)}{dt}$ et $q(t) = C.u_C(t)$

donc $i(t) = C.\frac{du_C(t)}{dt}$ soit $R.C.\frac{du_C(t)}{dt} + u_C(t) = E$

D'où :

$$\frac{du_C(t)}{dt} + \frac{1}{R.C}.u_C(t) = \frac{E}{R.C}$$

On dit que $u_C(t)$ satisfait à une équation différentielle non homogène du premier ordre.

Avec $q(t) = C.u_C(t)$, on a :

$$\frac{dq(t)}{dt} + \frac{1}{R.C}.q(t) = \frac{E}{R}$$

A t = 0, le condensateur est déchargé et $q(0) = C.u_C(0) = 0$, on en déduit : $\left.\frac{du_C}{dt}\right)_{t=0} = \frac{E}{R.C}$

On a donc un point de la courbe représentative de $u_C(t)$: (0 ; 0) ainsi que la valeur de la pente de la tangente à cette courbe à l'origine des dates.

Au bout d'un temps assez long ($t = \infty$) on peut considérer que le condensateur est chargé et qu'il ne passe plus de charge dans le circuit : $\left.\frac{dq}{dt}\right)_{t=\infty} = C.\left.\frac{du_C}{dt}\right)_{t=\infty} = 0$ donc $\left.\frac{du_C}{dt}\right)_{t=\infty} = 0$

De l'équation différentielle, on tire : $u_C(\infty) = E$ tension aux bornes du générateur.

La courbe représentative de $u_C(t)$ tend vers la valeur E qui représente une asymptote avec une pente nulle. La courbe tend exponentiellement vers cette valeur.

Les solutions de l'équation différentielle sont de la forme :

$u_C(t) = A.e^{-m.t} + B$ où $m > 0$, m et B constantes d'intégration, A constante non définie.

Introduisons cette expression dans l'équation : $\frac{d(A.e^{-m.t}+B)}{dt} + \frac{1}{R.C}.(A.e^{-m.t} + B) = \frac{E}{R.C}$

D'où : $-m.A.e^{-m.t} + \frac{A}{R.C}.e^{-m.t} + \frac{B}{R.C} = \frac{E}{R.C}$

Cette équation doit être vérifiée à chaque instant, on en déduit : B = E

D'où : $-m.A.e^{-m.t} + \frac{A}{R.C}.e^{-m.t} = 0$ et $m = \frac{1}{R.C}$

La solution générale de l'équation différentielle s'écrit : $u_C(t) = A.e^{-\frac{t}{R.C}} + E$

A est une constante qui dépend des <u>conditions initiales</u>.

Ici, à t = 0, le condensateur est déchargé, donc : $u_C(0) = 0 = A.e^{-0} + E$ d'où $A = -E$

Compte tenu des conditions initiales imposées par l'expérience, la solution est :

$$u_C(t) = E.(1 - e^{-\frac{t}{R.C}})$$

2. Décharge du condensateur dans une résistance

Le condensateur étant chargé, on bascule, à l'instant t=0, l'interrupteur en position 2.

A chaque instant $t > 0$, on a : $V_A - V_B = u_R(t) + u_C(t) = R.i(t) + u_C(t) = 0$

On a $i(t) = \frac{dq(t)}{dt}$ et $q(t) = C.u_C(t)$ donc $i(t) = C.\frac{du_C(t)}{dt}$ soit $R.C.\frac{du_C(t)}{dt} + u_C(t) = 0$

D'où l'équation : $$\frac{du_C(t)}{dt} + \frac{1}{R.C}.u_C(t) = 0$$

On dit que $u_C(t)$ satisfait à une équation différentielle homogène du premier ordre.

Avec $q(t) = C.u_C(t)$, on a $$\frac{dq(t)}{dt} + \frac{1}{R.C}.q(t) = 0$$

A $t = 0$, le condensateur est chargé $q(0) = C.u_C(0) = C.E$, on en déduit : $\left(\frac{du_C}{dt}\right)_{t=0} = -\frac{E}{R.C}$

On a donc un point de la courbe représentative de $u_C(t)$: (0 ; E) ainsi que la valeur de la pente de la tangente à cette courbe à l'origine des dates.

Au bout d'un temps assez long ($t = \infty$) on peut considérer que le condensateur est déchargé et qu'il ne passe plus de charge dans le circuit : $\left(\frac{dq}{dt}\right)_{t=\infty} = C.\left(\frac{du_C}{dt}\right)_{t=\infty} = 0$ donc

$\left(\frac{du_C}{dt}\right)_{t=\infty} = 0$ et $u_C(\infty) = 0$.

La courbe représentative de $u_C(t)$ tend vers 0 qui représente une asymptote avec une pente nulle. La courbe tend exponentiellement vers 0.

Les solutions de l'équation différentielle sont de la forme :

$u_C(t) = A.e^{-m.t} + B$ où m > 0, m et B constantes d'intégration, A constante non définie.

Introduisons cette expression dans l'équation : $\frac{d(A.e^{-m.t}+B)}{dt} + \frac{1}{R.C}.(A.e^{-m.t} + B) = 0$

D'où : $$- m.A.e^{-m.t} + \frac{A}{R.C}.e^{-m.t} + \frac{B}{R.C} = 0$$

Cette équation doit être vérifiée à chaque instant, on en déduit : B = 0

D'où : $$- m.A.e^{-m.t} + \frac{A}{R.C}.e^{-m.t} = 0 \text{ et } m = \frac{1}{R.C}$$

La solution générale de l'équation différentielle s'écrit : $u_C(t) = A.e^{-\frac{t}{R.C}}$

Là encore A est une constante qui dépend des <u>conditions initiales</u>.

Ici, à t = 0, le condensateur est chargé, donc : $u_C(0) = E = A.e^{-0}$ d'où A = E

Compte tenu des conditions initiales imposées par l'expérience, la solution est :

$$\boxed{u_C(t) = E.e^{-\frac{t}{R.C}}}$$

En appliquant les relations : $q(t) = C.u_C(t)$ et $i(t) = C.\frac{du_C(t)}{dt}$

On a : $$\boxed{q(t) = C.E.\ e^{-\frac{t}{R.C}} \text{ et } i(t) = -\frac{E}{R}.e^{-\frac{t}{R.C}}}$$

Au bout d'un temps long : $q(\infty) = 0$ le condensateur est déchargé

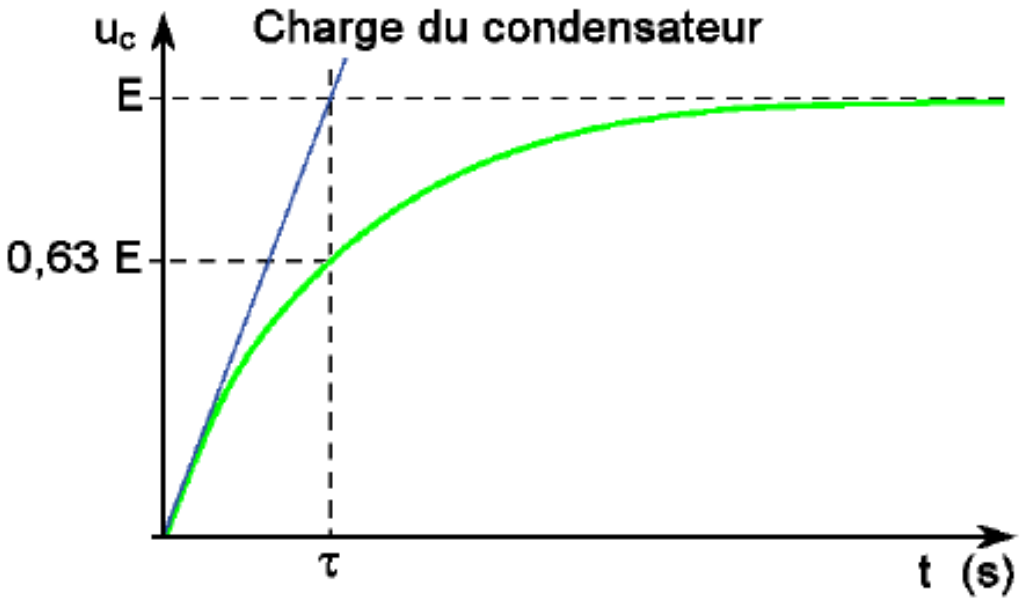
Et : $i(\infty) = 0$ il n'y a plus de courant

3. Constante de temps du dipôle RC

Le produit **R.C** est homogène à un temps et est appelé constante de temps τ du dipôle RC.

. Lors de la charge :

$$u_C(t) = E.(1 - e^{-\frac{t}{R.C}})$$



La tangente à la courbe à l'origine coupe l'asymptote y = E au point d'abscisse t = τ.
En effet, la tangente à la courbe représentative de $u_C(t)$, à l'origine des dates, a pour équation :

$$y = \left.\frac{d[u_C(t)]}{dt}\right)_{t=0} .t = - E.(- \frac{1}{R.C}).t = \frac{E}{R.C} .t$$

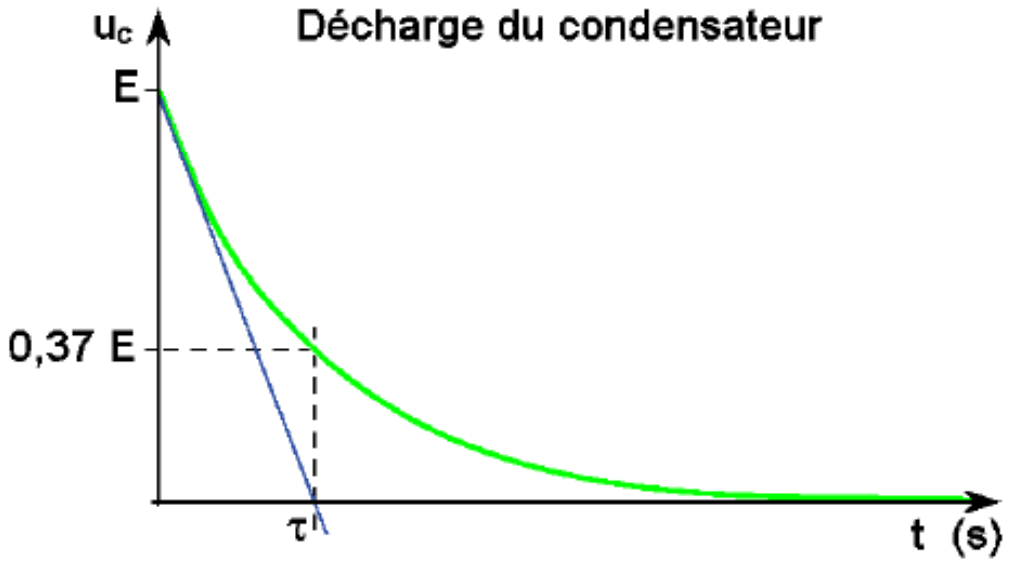
Elle coupe l'asymptote y = E en un point d'abscisse t : E/(R.C).t = E soit t = R.C = τ

Et : $u_C(\tau) = E.(1 - e^{-1}) \approx 0,63.E.$

Par lecture graphique de l'abscisse du point de la courbe dont l'ordonnée est égale à 0,63.E, on obtient la valeur de τ.

. **Lors de la décharge :**

$$u_C(t) = E.e^{-\frac{t}{R.C}}$$



La tangente à la courbe, à t=0s, coupe l'asymptote y=0 au point d'abscisse t = τ.

En effet, la tangente à la courbe représentative de $u_c(t)$, à t = 0s, a pour équation :

$$y = E + \left.\frac{d[u_C(t)]}{dt}\right)_{t=0}.t = E - \frac{E}{R.C}.t$$

Si t = τ = R.C , on a alors : y = 0

On a également :

$$u_C(\tau) = E.e^{-1} \approx 0,37.E$$

Par lecture graphique de l'abscisse du point de la courbe dont l'ordonnée est égale à 0,37.E, on obtient la valeur de τ.

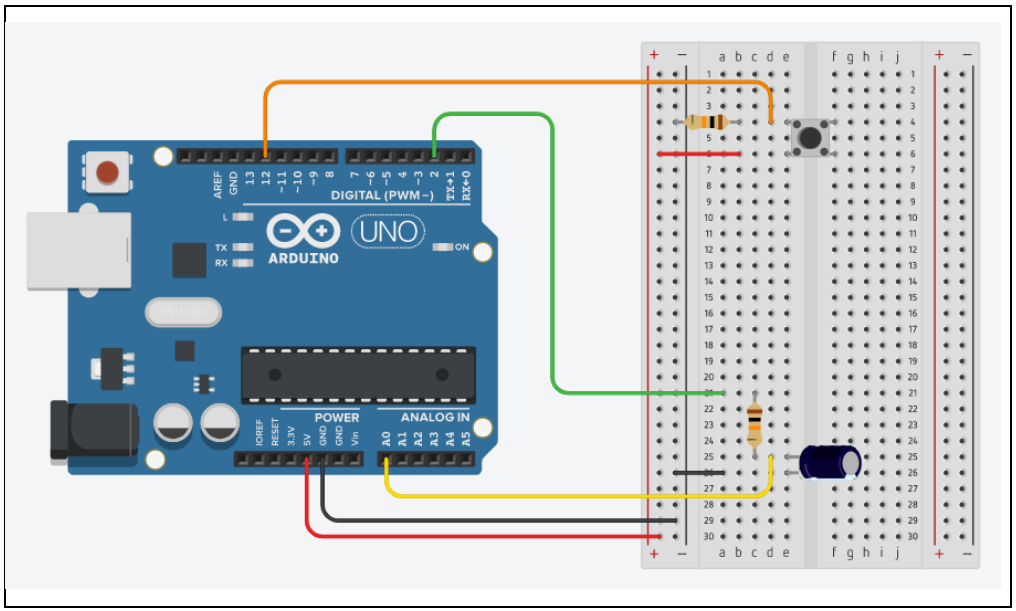
Activité 1 : Etude de la charge d'un condensateur d'un dipôle RC

L'objectif de l'activité est de suivre l'évolution temporelle de la tension aux bornes du condensateur lors de sa charge afin de vérifier la relation :

$$u_C(t) = E.(1 - e^{-\frac{t}{R.C}})$$

. Le circuit

L'activité sera réalisée avec le circuit suivant :



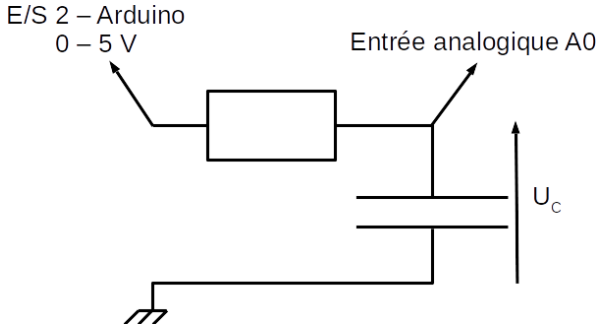
. Liste des composants :

. 1 condensateur de 100 µF (C chimique : **attention à la polarité**)
. 2 résistances de 10 kΩ (résistance du bouton poussoir et du dipôle RC)
. 1 bouton poussoir

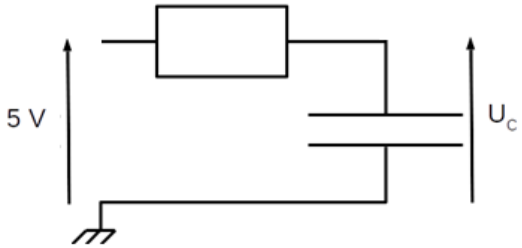
Dans ce montage, La carte Arduino est utilisée :

. pour appliquer une tension de 5 volts aux bornes du dipôle résistance - condensateur grâce à une sortie numérique (broche 2),

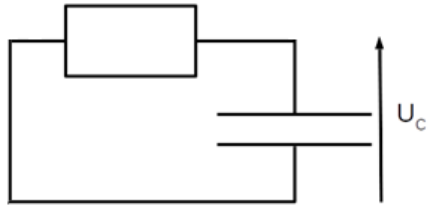
. pour mesurer la tension aux bornes du condensateur à l'aide d'une entrée analogique (broche A0).



Suivant l'état logique de broche N°2 déclarée en sortie numérique, le schéma électrique sera équivalent à :



Broche N°2 à niveau haut



Broche N°2 à niveau bas

Avec un Arduino, pour obtenir des mesures suffisamment précises, il faut utiliser un dipôle RC conduisant à des constantes de temps longues (quelques centaines de millisecondes au minimum).

. Descriptif de l'activité

Après avoir déchargé le condensateur, la mesure de la tension aux bornes du condensateur U_c, lors de la charge, à l'aide de l'entrée analogique A0 est lancée, à t = 0 s, par un appui sur le bouton poussoir.

La valeur de la tension en V est affichée dans le moniteur série toutes les 100 ms.

Les mesures sont arrêtées en appuyant sur le bouton poussoir. Le condensateur est alors déchargé afin de pouvoir effectuer de nouvelles mesures en appuyant de nouveau sur le bouton poussoir.

Il est donc possible d'acquérir des couples de données (t, U_c) afin de vérifier la relation U_c=f(t) théorique.

. Le programme

Voici le code de l'activité (**"Activity1.ino" dans le dossier "Codes/Circuit RC"**) :

Activity1

```
// Déclaration des constantes et variables

const int PinUC = 0;
const int PinButton = 12;
const int PinAlimC = 2;

int ValPinUC = 0;
float UC = 0.0;
unsigned long t0;
float dt;

int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;


// Déclaration de fonctions

void decharge(){
  digitalWrite(PinAlimC, LOW);
  Serial.println("Decharge du condensateur");
  while (analogRead(PinUC) > 0) {
    delay(200);
    }
  Serial.println("Condensateur decharge");
}


// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton, INPUT);
  pinMode(PinAlimC, OUTPUT);
  decharge();
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}
```

```
// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
delay(10);

if ((ValButton == HIGH)&&(OldValButton == LOW))
  {
  State=1-State;
  }
OldValButton = ValButton;

if (State==1)
  {
  if (OldState == 0)
  {
   Serial.println("Charge du condensateur en cours.");
   Serial.println("");
   Serial.println ("Temps (S);Uc (V):");
   t0 = micros();
   digitalWrite(PinAlimC, 1);
   OldState=1;
  }
  ValPinUC = analogRead(PinUC);
  UC = (ValPinUC/1023.0)*5.0;
  dt =  (micros() - t0)* 1e-6;
  Serial.print(dt,2);
  Serial.print(";");
  Serial.println(UC,2);

  delay(100);
  }
  else
  {
    if (OldState == 1){
    Serial.println("Fin des mesures.");
    decharge();
    OldState = 0;}
  }
}
```

Déroulement du programme :

- 1. Déclaration des constantes et variables :

. **const int PinUc = 0** (broche du condensateur : A0)
. **const int PinButton = 12** (broche du bouton poussoir)
. **const int PinAlimC = 2** (broche d'alimentation du dipôle RC)

. **int ValPinUc = 0** (variable nombre entier valeur broche du condensateur)
. **float Uc = 0.0** (variable nombre décimal calcul tension Uc)
. **unsigned long t0** (variable nombre entier long temps début charge)
. **float dt** (variable nombre décimal différence de temps entre les mesures)

. **int ValButton = 0** (variable nombre entier valeur broche bouton poussoir)
. **int OldValButton = 0** (variable nbr entier ancienne valeur broche btn poussoir)
. **int State = 0** (variable nombre entier pour action à effectuer)
. **int OldState = 0** (variable nombre entier pour action effectuée précédemment)

- 2. Déclaration de fonctions :

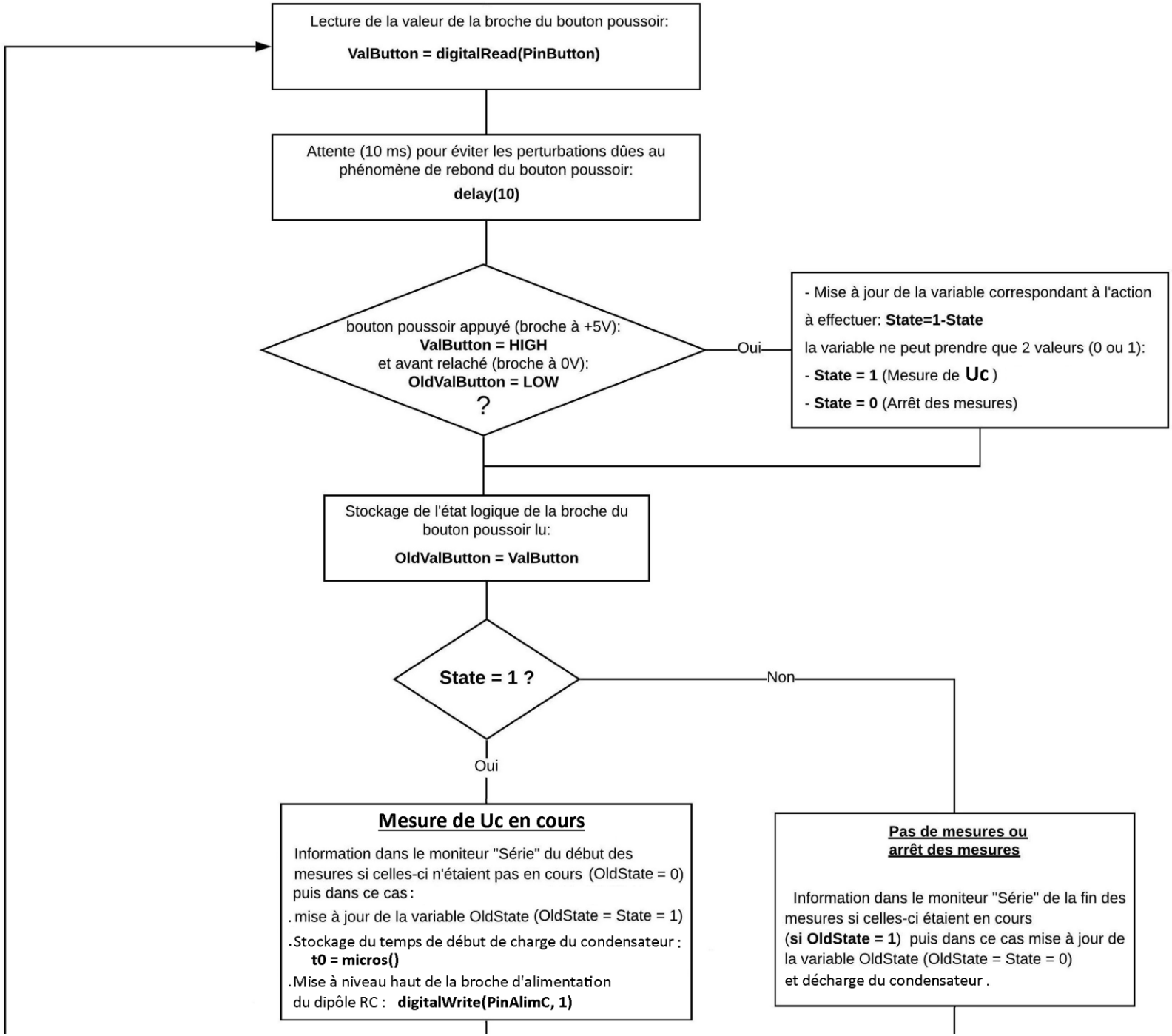
→ Fonction permettant de décharger le condensateur :

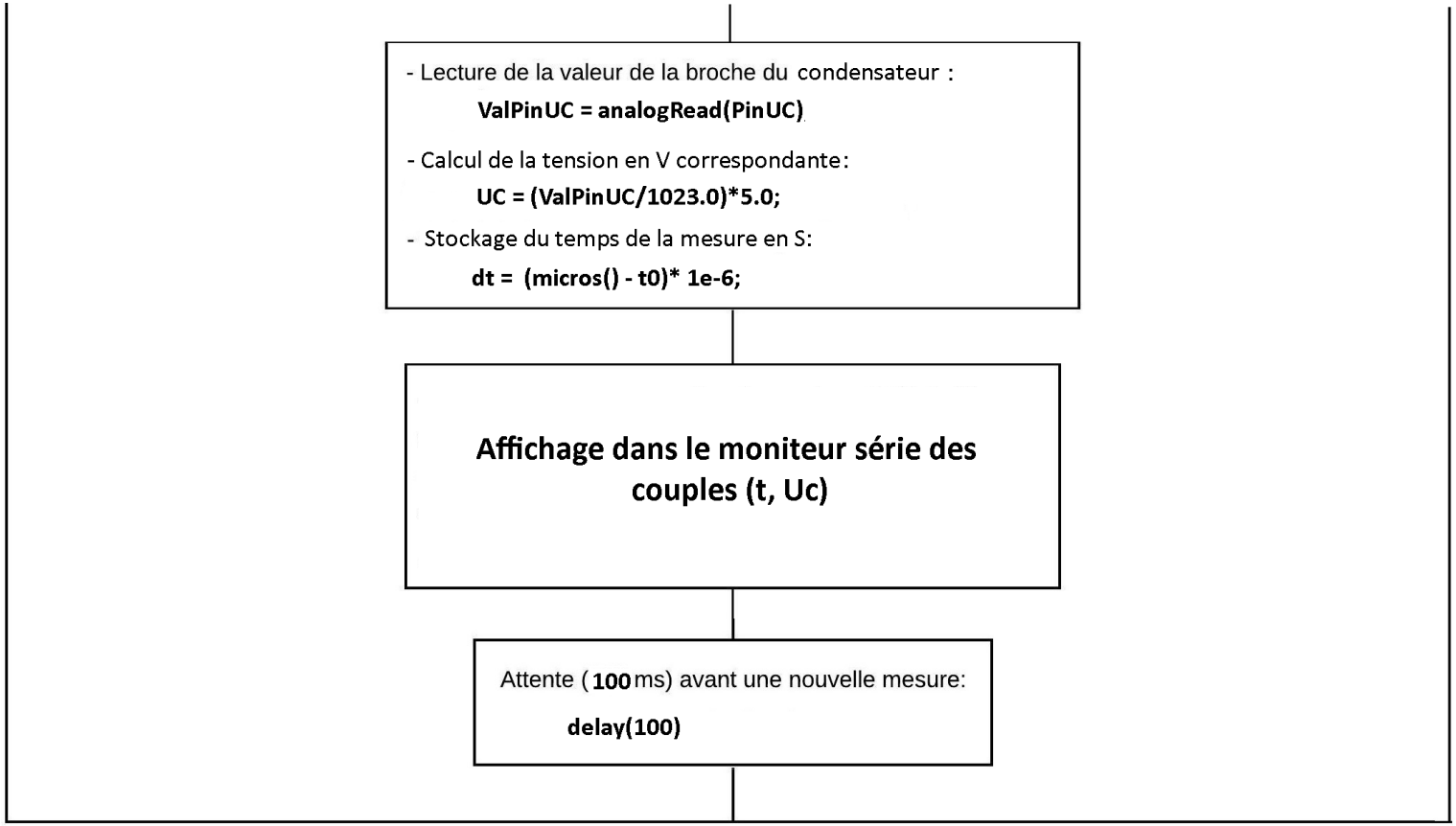
- Mise à niveau bas de la broche d'alimentation du dipôle RC : **digitalWrite(PinAlimC, LOW)**
- Attente de la fin de la décharge du condensateur : **while (analogRead(PinUC) > 0)**)

- 3. Initialisation des entrées et sorties :

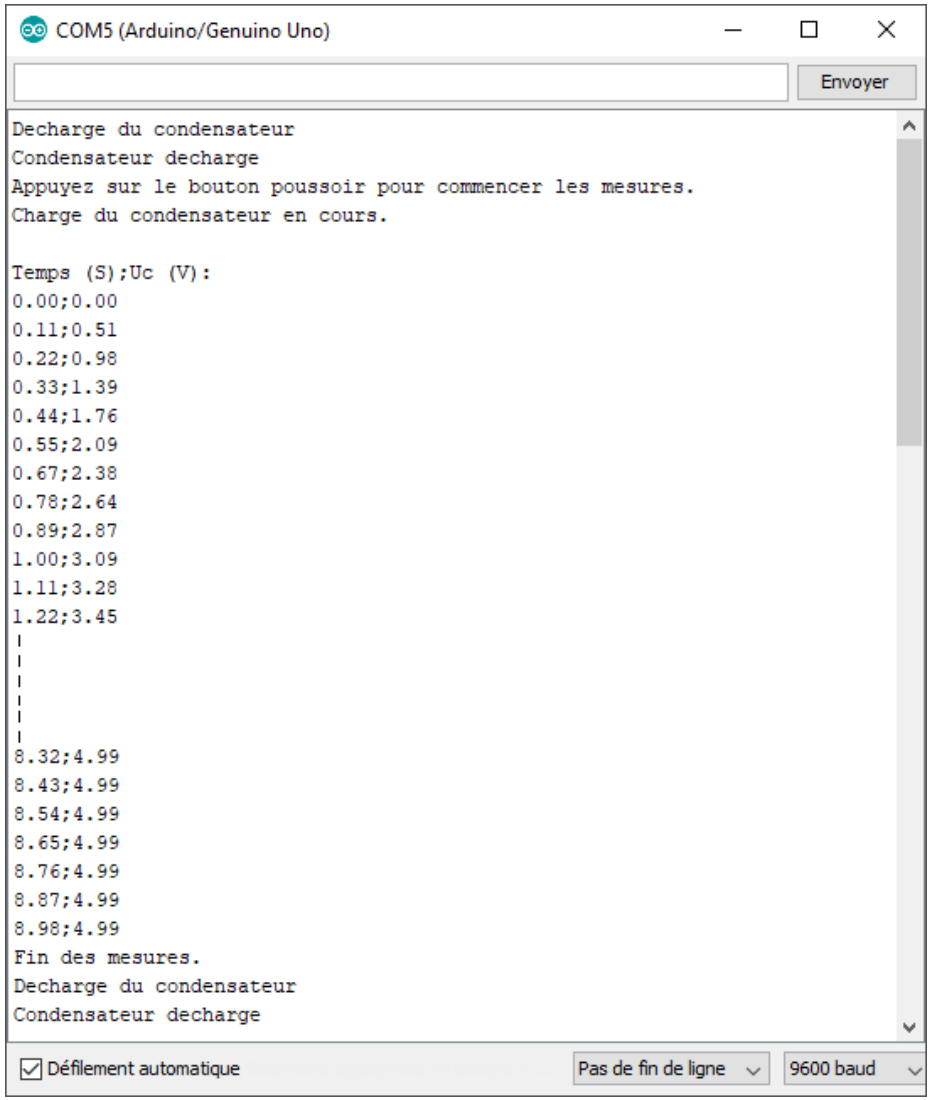
. Initialisation de la liaison série à un débit de 9600 bauds,
. Initialisation de la broche du bouton poussoir en entrée,
. Initialisation de la broche d'alimentation du dipôle RC en sortie,
. Décharge du condensateur.

- 4. Fonction principale en boucle :


Lecture de la valeur de la broche du bouton poussoir:
ValButton = digitalRead(PinButton)
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
delay(10)
bouton poussoir appuyé (broche à +5V):
ValButton = HIGH
et avant relaché (broche à 0V):
OldValButton = LOW
?
Oui
- Mise à jour de la variable correspondant à l'action à effectuer: State=1-State
la variable ne peut prendre que 2 valeurs (0 ou 1):
- State = 1 (Mesure de Uc)
- State = 0 (Arrêt des mesures)
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de Uc en cours
Information dans le moniteur "Série" du début des mesures si celles-ci n'étaient pas en cours (OldState = 0) puis dans ce cas :
. mise à jour de la variable OldState (OldState = State = 1)
. Stockage du temps de début de charge du condensateur : t0 = micros()
. Mise à niveau haut de la broche d'alimentation du dipôle RC : digitalWrite(PinAlimC, 1)
Pas de mesures ou arrêt des mesures
Information dans le moniteur "Série" de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0) et décharge du condensateur .


- Lecture de la valeur de la broche du condensateur :
ValPinUC = analogRead(PinUC)
- Calcul de la tension en V correspondante:
UC = (ValPinUC/1023.0)*5.0;
- Stockage du temps de la mesure en S:
dt = (micros() - t0)* 1e-6;
Affichage dans le moniteur série des couples (t, Uc)
Attente (100 ms) avant une nouvelle mesure:
delay(100)

Résultats dans le moniteur série :



Remarque :

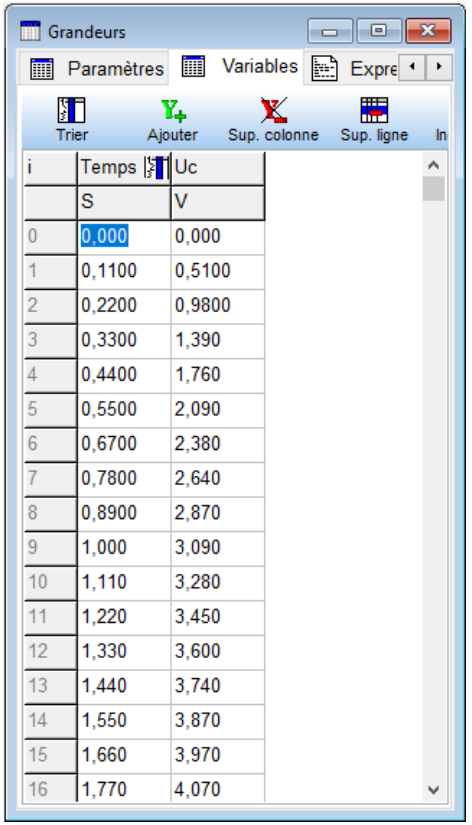
La fonction **"micros()"** renvoie le nombre de microsecondes, sous la forme d'un nombre de type unsigned long, depuis que la carte Arduino a démarré le programme en cours. Ce nombre repasse à 0 après approximativement 70 minutes.

Syntaxe :

variable_unsigned_long = micros();

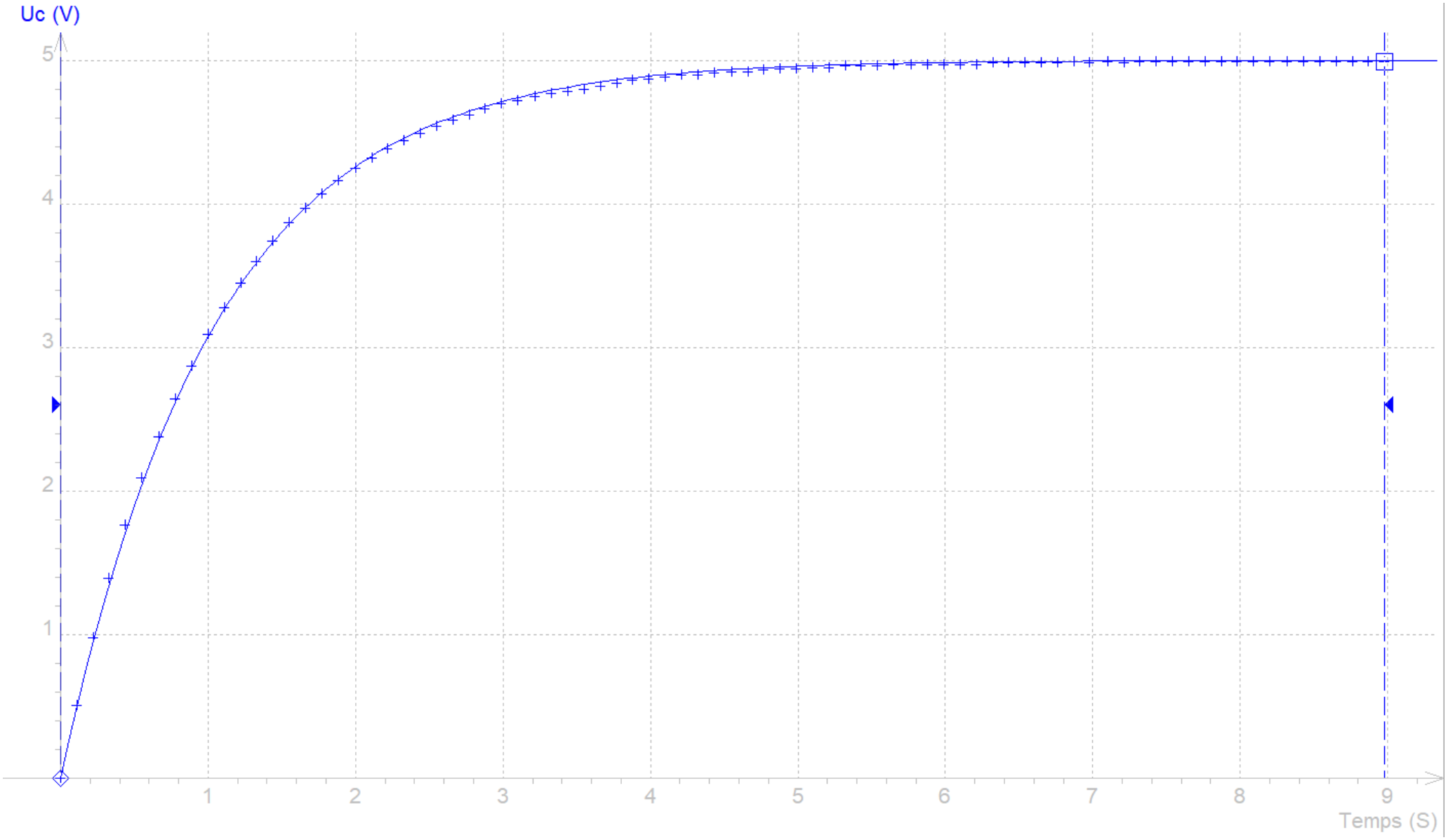
Exploitation des résultats :

Pour exploiter les mesures, il suffit de sélectionner et de copier toutes les données à partir du moniteur série et d'ouvrir un nouveau fichier dans Regressi à partir du "Presse-papier**".**



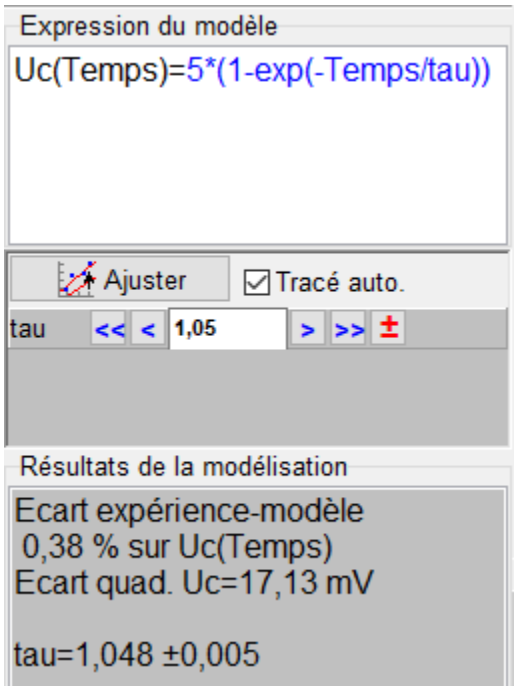
i	Temps	Uc
	S	V
0	0,000	0,000
1	0,1100	0,5100
2	0,2200	0,9800
3	0,3300	1,390
4	0,4400	1,760
5	0,5500	2,090
6	0,6700	2,380
7	0,7800	2,640
8	0,8900	2,870
9	1,000	3,090
10	1,110	3,280
11	1,220	3,450
12	1,330	3,600
13	1,440	3,740
14	1,550	3,870
15	1,660	3,970
16	1,770	4,070

On peut alors tracer le graphe représentant la tension aux bornes du condensateur en fonction du temps :



Et effectuer une modélisation suivant :

$$u_C(t) = E.(1 - e^{-\frac{t}{R.C}})$$

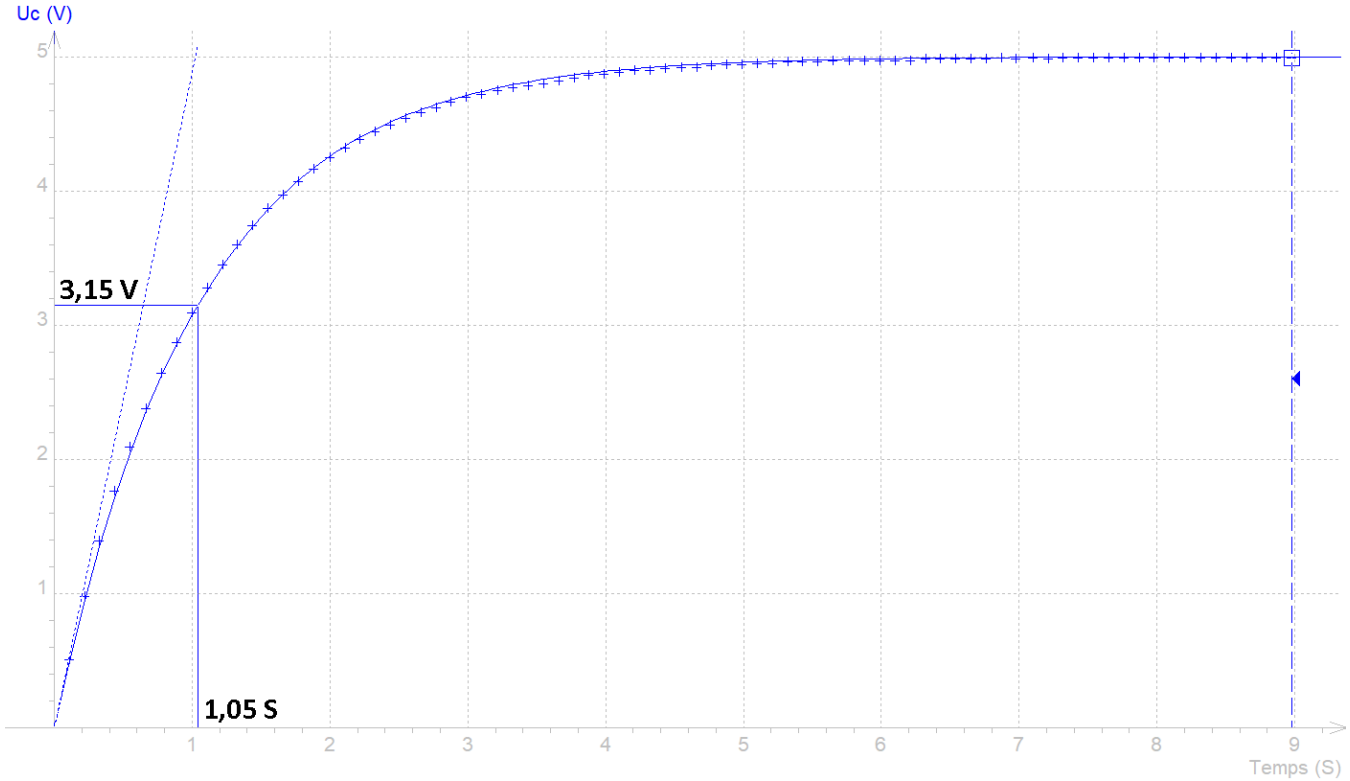


La valeur théorique de τ est :

$$\tau = RC = 10.10^3 \times 100.\ 10^{-6} = 1\ S$$

Par la modélisation, la détermination de τ donne une valeur très proche de la valeur théorique.

On peut également déterminer τ à l'aide de la tangente à l'origine ou par la mesure du temps pour lequel le condensateur est chargé à 63 % (0,63 x 5 = 3,15 V):



Activité 2 : Etude de la décharge d'un condensateur d'un dipôle RC

L'objectif de l'activité est de suivre l'évolution temporelle de la tension aux bornes du condensateur lors de sa décharge, avec le même circuit que l'activité précédente, afin de vérifier la relation :

$$u_C(t) = E.e^{-\frac{t}{R.C}}$$

. Descriptif de l'activité

Après avoir chargé le condensateur, la mesure de la tension aux bornes du condensateur U_c, lors de la décharge, à l'aide de l'entrée analogique A0 est lancée, à t=0s, par un appui sur le bouton poussoir.

La valeur de la tension en V est affichée dans le moniteur série toutes les 100 ms.

Les mesures sont arrêtées en appuyant sur le bouton poussoir. Le condensateur est alors chargé afin de pouvoir effectuer de nouvelles mesures en appuyant de nouveau sur le bouton poussoir.

Il est donc possible d'acquérir des couples de données (t, U_c) afin de vérifier la relation U_c=f(t) théorique.

. Le programme

Voici le code de l'activité (**"Activity2.ino" dans le dossier "Codes/Circuit RC"**) :

Activity2

```
// Déclaration des constantes et variables

const int PinUC = 0;
const int PinButton = 12;
const int PinAlimC = 2;

int ValPinUC = 0;
float UC = 0.0;
unsigned long t0;
float dt;

int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;


// Déclaration de fonctions

void charge(){
  digitalWrite(PinAlimC, HIGH);
  Serial.println("Charge du condensateur");
  while (analogRead(PinUC) < 4.99) {
    delay(200);
    }
  Serial.println("Condensateur charge");
}


// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton, INPUT);
  pinMode(PinAlimC, OUTPUT);
  charge();
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.")
}
```

```
// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
delay(10);

if ((ValButton == HIGH)&&(OldValButton == LOW))
  {
  State=1-State;
  }
OldValButton = ValButton;

if (State==1)
  {
  if (OldState == 0)
  {
   Serial.println("Decharge du condensateur en cours.");
   Serial.println("");
   Serial.println ("Temps (S);Uc (V):");
   t0 = micros();
   digitalWrite(PinAlimC, 0);
   OldState=1;
  }
  ValPinUC = analogRead(PinUC);
  UC = (ValPinUC/1023.0)*5.0;
  dt =  (micros() - t0)* 1e-6;
  Serial.print(dt,2);
  Serial.print(";");
  Serial.println(UC,2);

  delay(100);
  }
  else
  {
    if (OldState == 1){
    Serial.println("Fin des mesures.");
    charge();
    OldState = 0;}
  }
}
```

Déroulement du programme :

- 1. Déclaration des constantes et variables :

Idem Activité 1

- 2. Déclaration de fonctions :

→ Fonction permettant de charger le condensateur :

- Mise à niveau haut de la broche d'alimentation du dipôle RC : **digitalWrite(PinAlimC, LOW)**
- Attente de la fin de la charge du condensateur : **while (analogRead(PinUC) < 4.99)**)

- 3. Initialisation des entrées et sorties :

. Initialisation de la liaison série à un débit de 9600 bauds,
. Initialisation de la broche du bouton poussoir en entrée,
. Initialisation de la broche d'alimentation du dipôle RC en sortie,
. Charge du condensateur.

- 4. Fonction principale en boucle :

Idem Activité 1, excepté le fait que la broche d'alimentation du condensateur soit mise à un niveau bas pour pouvoir décharger le condensateur.

Et à la fin des mesures, le condensateur est rechargé, afin de pouvoir renouveler l'expérience.

Résultats dans le moniteur série :

COM5 (Arduino/Genuino Uno)

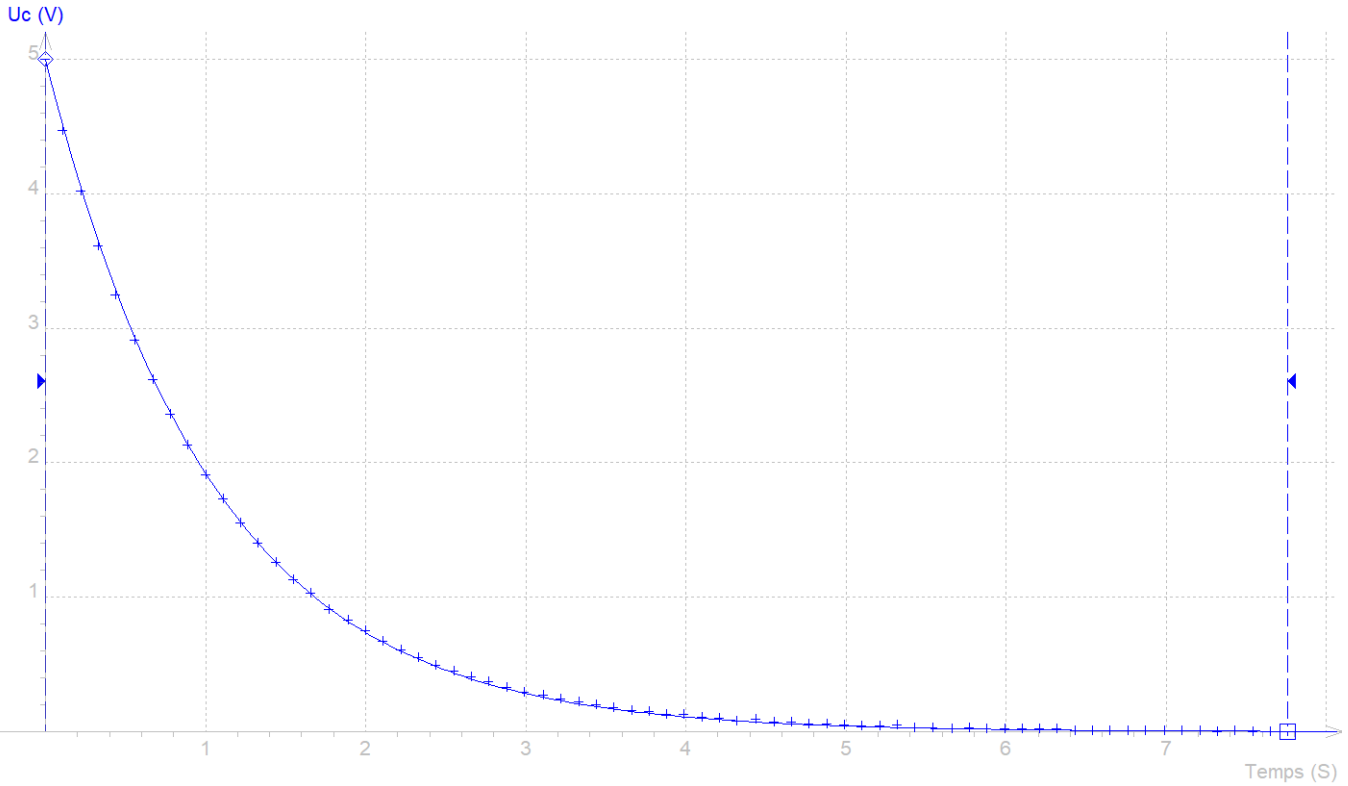
```
Charge du condensateur
Condensateur charge
Appuyez sur le bouton poussoir pour commencer les mesures.
Decharge du condensateur en cours.

Temps (S);Uc (V):
0.00;5.00
0.11;4.47
0.22;4.02
0.33;3.61
0.44;3.25
0.56;2.91
0.67;2.62
0.78;2.36
0.89;2.13
1.00;1.91
1.11;1.73
  |
  |
  |
  |
6.99;0.01
7.10;0.01
7.21;0.01
7.32;0.00
7.43;0.01
7.54;0.00
7.65;0.00
7.76;0.00
Fin des mesures.
Charge du condensateur
Condensateur charge
```

Défilement automatique | Pas de fin de lig

Exploitation des résultats :

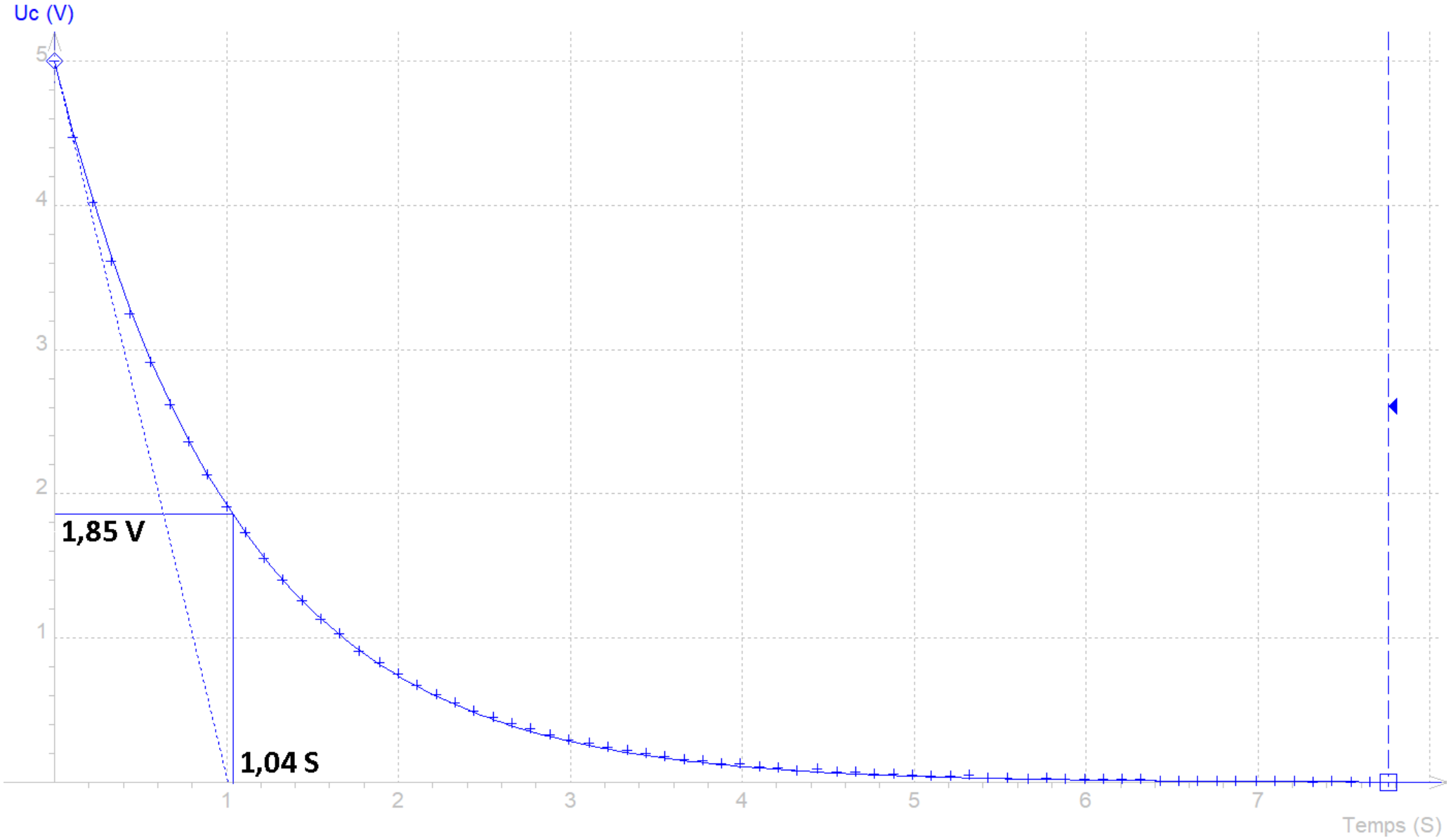
De même que précédemment, on utilise Regressi pour effectuer la modélisation de la représentation graphique de Uc=f(t) :



Expression du modèle	Résultats de la modélisation
Uc(Temps)=5*exp(-Temps/tau)	Ecart expérience-modèle 0,89 % sur Uc(Temps) Ecart quad. Uc=12,1 mV tau=1,045 ±0,003

Par la modélisation, la détermination de τ donne également une valeur très proche de la valeur théorique.

On peut également déterminer τ à l'aide de la tangente à l'origine ou par la mesure du temps pour lequel le condensateur est déchargé à 37 % (0,37 x 5 = 1,85 V):



Activité 3 : Détermination de la capacité d'un condensateur par mesure d'une constante de temps

Dans cette activité, nous allons modifier le programme de l'activité 1 (étude de la charge d'un condensateur) de façon cette fois-ci à mesurer directement la constante de temps du circuit RC sans passer par le tracé de la caractéristique Uc=f(t).

On pourra alors déterminer la capacité d'un condensateur inconnu puisque :

$\tau = RC$ donc : $C = \tau / R$

. Descriptif de l'activité

Lors de la charge d'un condensateur, nous avons vu, qu'au bout d'un temps égal à la constante de temps τ, le condensateur était chargé à 63 % de la tension appliquée au dipôle RC (ici, à t = τ : Uc= 0,63 x 5 = 3,15 V).

Avec le même circuit que les activités précédentes, pour déterminer la constante de temps, il suffit donc, par lecture de l'entrée analogique A0, de mesurer le temps que met cette entrée pour atteindre la valeur correspondant à la tension aux bornes du condensateur au temps τ, soit : **0,63 x 1023 = 644** (CAN 5 V = 1023).

Ainsi, après avoir déchargé le condensateur, la mesure de la tension aux bornes du condensateur U_c, lors de la charge, à l'aide de l'entrée analogique A0 est lancée, à t = 0 s, par un appui sur le bouton poussoir.

Quand la valeur de l'entrée analogique A0 souhaitée est atteinte, la constante de temps est alors calculée et affichée dans le moniteur série, puis le condensateur est déchargé.

Une nouvelle mesure est possible en appuyant de nouveau sur le bouton poussoir.

. Le programme

Voici le code de l'activité (**"Activity3.ino" dans le dossier "Codes/Circuit RC"**) :

Activity3

```
// Déclaration des constantes et variables

const int PinUC = 0;
const int PinButton = 12;
const int PinAlimC = 2;

int ValPinUC = 0;
unsigned long t0;
float dt;

int ValButton = 0;
int OldValButton = 0;
int State = 0;


// Déclaration de fonctions

void decharge(){
  digitalWrite(PinAlimC, LOW);
  Serial.println("Decharge du condensateur.");
  while (analogRead(PinUC) > 0) {
    delay(200);
    }
  Serial.println("Condensateur decharge.");
}
```

```
// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton, INPUT);
  pinMode(PinAlimC, OUTPUT);
  decharge();
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}


// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
delay(10);

if ((ValButton == HIGH)&&(OldValButton == LOW))
  {
  State=1-State;
  }
OldValButton = ValButton;

if (State==1)
  {
  Serial.println("Charge du condensateur en cours.");
  Serial.println("");
  digitalWrite(PinAlimC, 1);
  t0 = micros();
  while (ValPinUC<644) {
    ValPinUC = analogRead(PinUC);
  }
  dt =  (micros() - t0)* 1e-6;
  Serial.print("Tau = ");
  Serial.println(dt,2);
  Serial.println("");
  Serial.println("Fin des mesures.");
  decharge();
  State = 0;
  ValPinUC = 0;
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
  }
}
```

Déroulement du programme :

- 1. Déclaration des constantes et variables :

. **const int PinUc = 0** (broche du condensateur)
. **const int PinButton = 12** (broche du bouton poussoir)
. **const int PinAlimC = 2** (broche d'alimentation du dipôle RC)

. **int ValPinUc = 0** (variable nombre entier valeur broche du condensateur)
. **unsigned long t0** (variable nombre entier long temps début charge)
. **float dt** (variable nombre décimal temps entre les mesures)

. **int ValButton = 0** (variable nombre entier valeur broche bouton poussoir)
. **int OldValButton = 0** (variable nbr entier ancienne valeur broche btn poussoir)
. **int State = 0** (variable nombre entier pour action à effectuer)

- 2. Déclaration de fonctions :

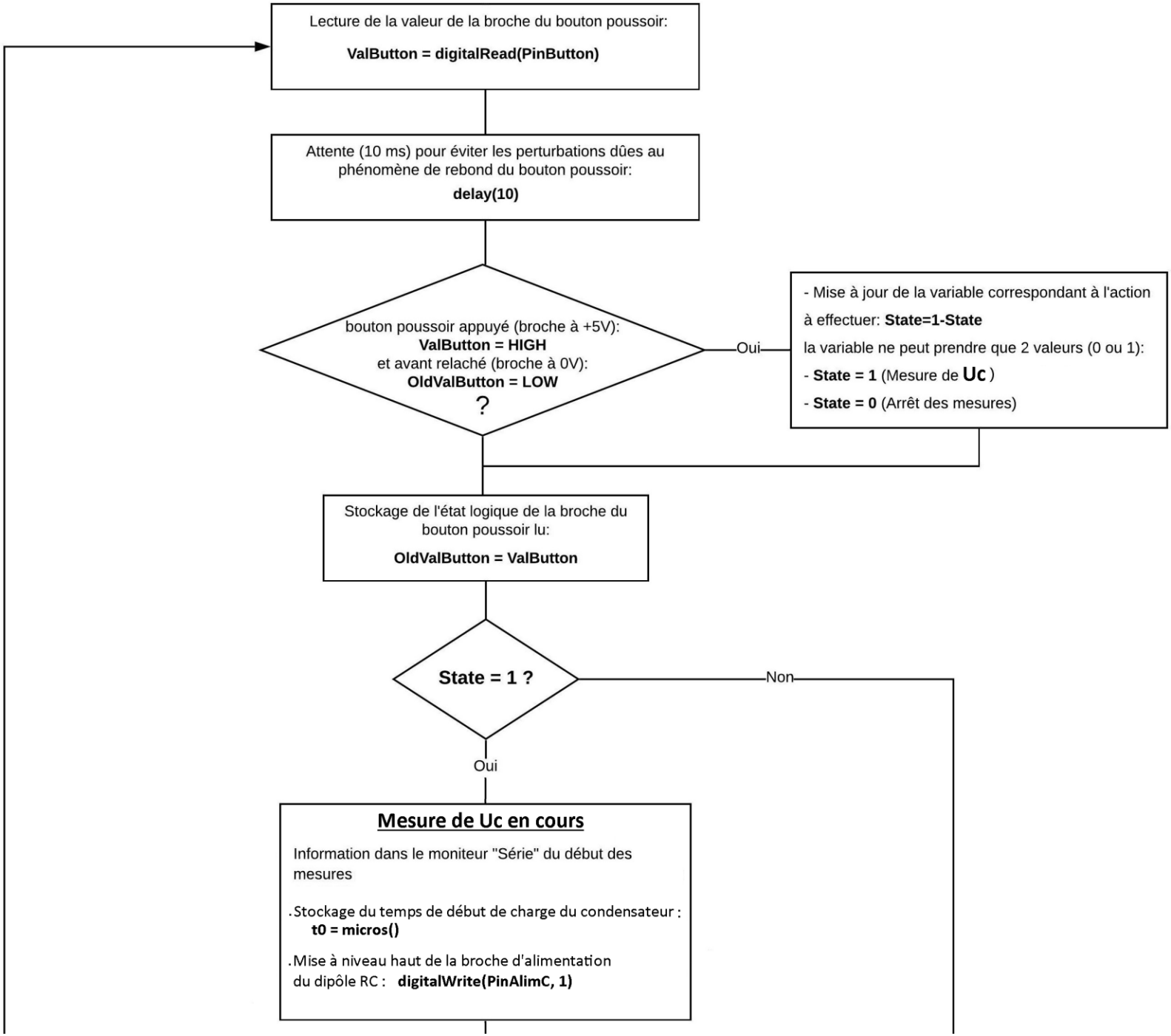
→ Fonction permettant de décharger le condensateur :

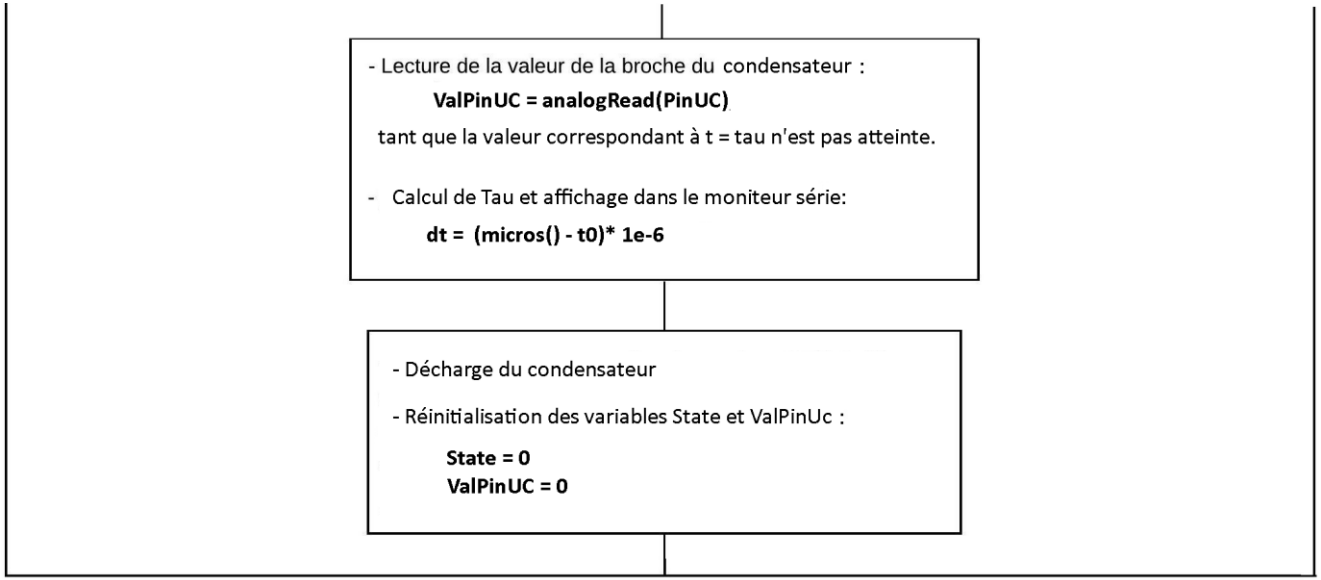
- Mise à niveau bas de la broche d'alimentation du dipôle RC : **digitalWrite(PinAlimC, LOW)**
- Attente de la fin de la décharge du condensateur : **while (analogRead(PinUC) > 0)**)

- 3. Initialisation des entrées et sorties :

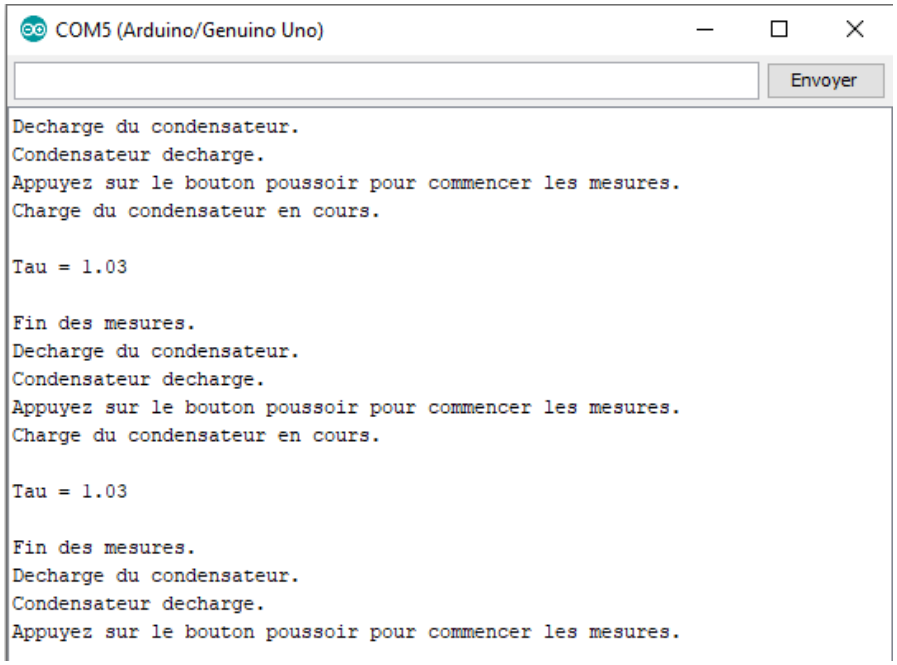
. Initialisation de la liaison série à un débit de 9600 bauds,
. Initialisation de la broche du bouton poussoir en entrée,
. Initialisation de la broche d'alimentation du dipôle RC en sortie,
. Décharge du condensateur.

- 4. Fonction principale en boucle :


Lecture de la valeur de la broche du bouton poussoir:
ValButton = digitalRead(PinButton)
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
delay(10)
bouton poussoir appuyé (broche à +5V):
ValButton = HIGH
et avant relaché (broche à 0V):
OldValButton = LOW
?
Oui
- Mise à jour de la variable correspondant à l'action à effectuer: State=1-State
la variable ne peut prendre que 2 valeurs (0 ou 1):
- State = 1 (Mesure de Uc)
- State = 0 (Arrêt des mesures)
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de Uc en cours
Information dans le moniteur "Série" du début des mesures
.Stockage du temps de début de charge du condensateur :
t0 = micros()
.Mise à niveau haut de la broche d'alimentation du dipôle RC : digitalWrite(PinAlimC, 1)



Résultats dans le moniteur série :



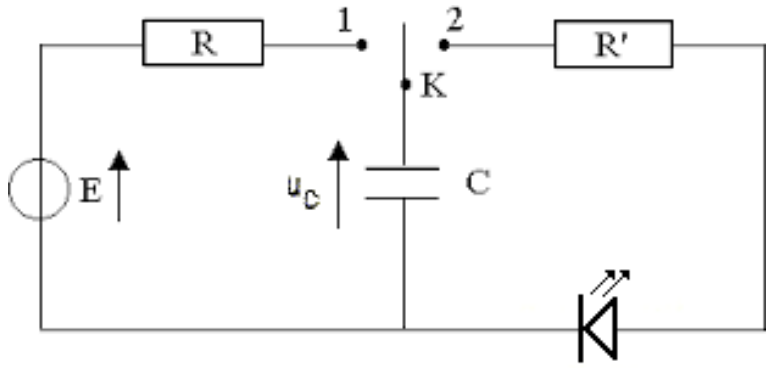
Les valeurs de τ sont conformes à celles obtenues lors de la première activité.

On peut alors déterminer la capacité du condensateur :

$$C = \tau/R = 1{,}03 / 10.10^{3} = 1{,}03.10^{-4}\ F = 103\ \mu F$$

Activité 4 : Simulation d'un flash photographique

L'objectif de cette activité est de simuler le flash d'un appareil-photo à l'aide d'une diode électroluminescente (DEL) blanche dans le circuit de décharge d'un condensateur selon le circuit suivant :



Dans ce circuit, quand l'interrupteur est en position 1, le condensateur se charge à travers la résistance R et quand l'interrupteur est en position 2, le condensateur se décharge dans la résistance R' et la DEL.

Pour simuler un flash, la DEL doit éclairer pendant une courte durée. La constante de temps du circuit de décharge doit donc être petite.

On prendra :

. R= 1 kΩ, R' = 100 Ω, C= 470 µF

. E = 5 V (Vcc Arduino)

. DEL blanche (tension de seuil = 2,5 V)

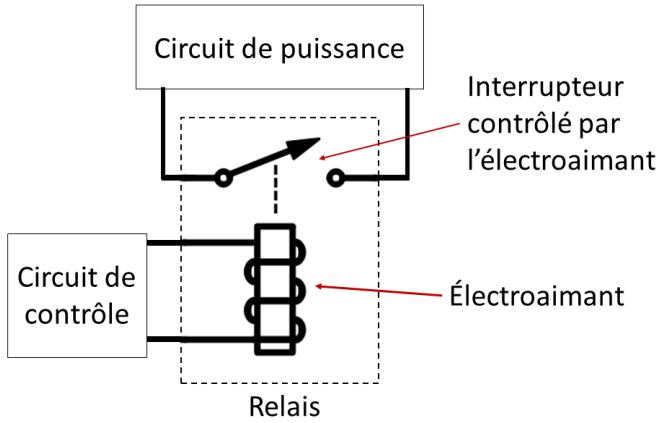
La constante de temps du circuit de décharge est alors :

$$\tau = R\,C = 100 \times 470.10^{-6} = 47\ mS$$

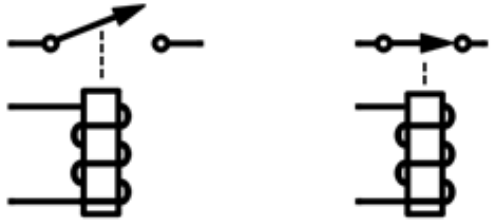
On utilisera un relais électromécanique, par exemple le SRS-06VDC-SL, pour jouer le rôle de l'interrupteur.

. Les relais

Un relais électromécanique, c'est un interrupteur commandé par un électroaimant. On l'utilise généralement pour isoler un circuit de commande (qui actionne l'électroaimant) d'un circuit de puissance (qui est contrôlé par l'interrupteur)



Le type de relais le plus simple est le relais SPST ("Single pole single throw"). Il est muni de 4 connecteurs : 2 connecteurs pour le contrôle de l'électroaimant, et deux connecteurs reliés à l'interrupteur.

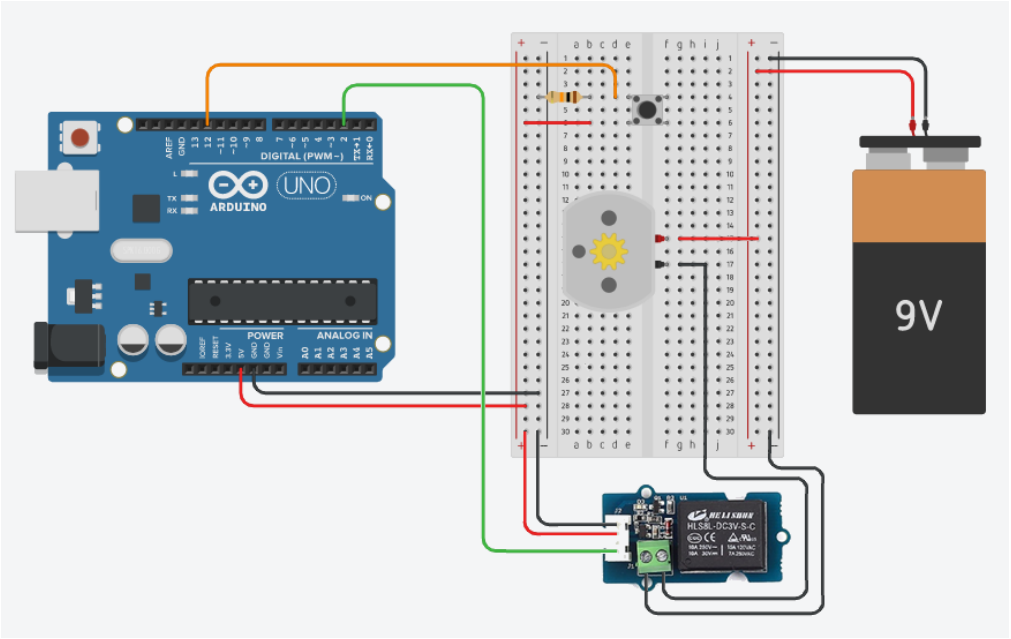


L'interrupteur bloque le courant du circuit de puissance si aucun courant ne circule dans l'électroaimant, et laisse circuler le courant si un courant circule dans l'électroaimant (on entend un petit "clic" lorsque l'interrupteur change d'état).

Par exemple, le relais **Grove 103020005** est un relais SPST qui agit comme un interrupteur normalement ouvert :

Il permet de commuter des charges plus élevées que ce que permettent les cartes Arduino. En effet, avec un relais, il est possible par exemple de contrôler un moteur fonctionnant sous 9 V continu.

Ce module est alimenté en 5V par l'Arduino et se raccorde (broche **"Signal")** sur une sortie digitale, comme dans le circuit suivant :



Avec ce circuit, quand la sortie digitale 2 de l'Arduino est à un niveau bas, le relais agit comme un interrupteur ouvert, le circuit du moteur est donc ouvert et le moteur ne tourne pas. Au contraire, quand la sortie digitale 2 est à un niveau haut, le relais fait contact, la DEL du module s'allume, le circuit du moteur est fermé et le moteur tourne.

C'est avec le bouton poussoir qu'on donnera l'ordre à l'Arduino d'allumer ou d'éteindre le moteur :

- bouton appuyé : le moteur tourne (sortie digitale 2 à HIGH)
- bouton relâché : le moteur est arrêté (sortie digitale 2 à LOW)

Voici le code permettant de contrôler un relais Grove avec un bouton poussoir (**"Relais.ino" dans le dossier "Codes/Circuit RC"**) :

Relais

```
// Déclaration des constantes et variables

const int buttonPin = 12;
const int relayPin =  2;


// Initialisation des entrées et sorties

void setup()
{
    pinMode(relayPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}


// Fonction principale en boucle

void loop()
{
    int buttonState = digitalRead(buttonPin);

    if (buttonState == 1)
    {
        digitalWrite(relayPin, HIGH);
    }
    else
    {
        digitalWrite(relayPin, LOW);
    }
    delay(10);
}
```

<u>Déroulement du programme</u> :

- 1. Déclaration des constantes et variables :

. **const int buttonPin = 12** (broche du bouton poussoir)

. **const int relayPin = 2** (broche du relais)

- 2. Initialisation des entrées et sorties :

. Initialisation de la broche du relais en sortie,

. Initialisation de la broche du bouton poussoir en entrée.

- 3. Fonction principale en boucle :

. Lecture de l'état logique de la broche du bouton poussoir

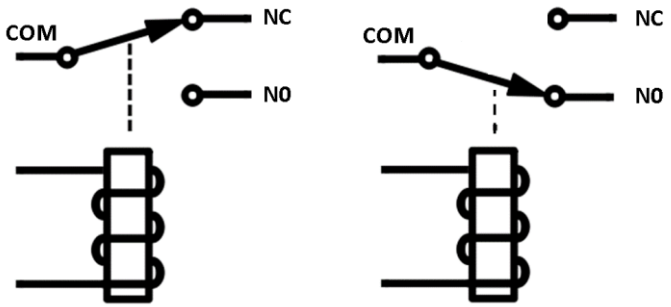
. Mise à jour de la valeur de la broche du relais en fonction de la valeur de la broche du bouton poussoir :

- bouton appuyé (buttonState =1) : broche du relais à HIGH

- bouton relâché (buttonState =0) : broche du relais à LOW

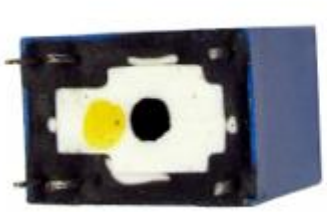
Cependant un relais SPST n'est pas adapté à notre activité. Nous allons plutôt utiliser un relais SPDT ("Single pole double throw")

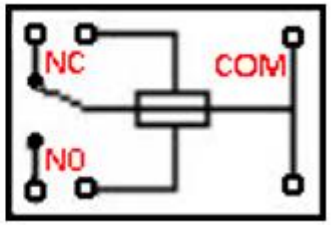
Dans ce type de relais, l'interrupteur est remplacé par un commutateur. Le relais comporte maintenant 5 connecteurs : en plus des deux connecteurs reliés à l'électroaimant, il y a un connecteur "COM" (commun), un connecteur "NC" (normally closed) et un connecteur "NO" (normally open).



En absence ou en présence de courant circulant dans l'électroaimant, le connecteur COM est en contact avec le connecteur NC ou le connecteur NO.

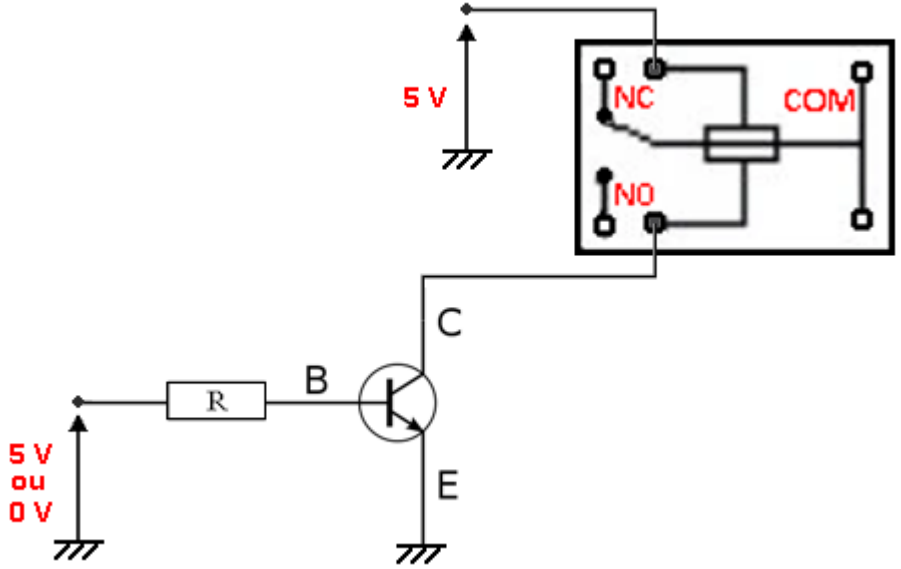
Voici le schéma de câblage du relais SRS-06VDC-SL que nous allons utiliser (vue de dessous en absence de courant dans la bobine) :





Contrairement au module Relais Grove, qui dispose déjà de son circuit de contrôle pour faire circuler ou pas un courant dans la bobine de commutation en fonction de la tension appliquée sur la broche **"Signal"**, le relais SRS-06VDC-SL, lui n'en dispose pas.

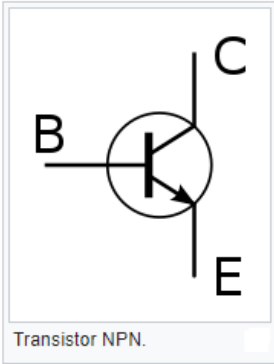
Il faut donc ajouter un circuit de contrôle à notre relais pour commuter de la position NC à NO. Pour cela, nous allons utiliser un transistor bipolaire NPN (BC547B) en mode interrupteur commandé avec le circuit suivant :



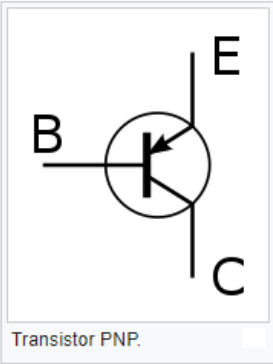
R = 2,2 kΩ

Les transistors bipolaires (BJT pour Bipolar Junction Transistor) sont des composants à trois broches, sur lesquelles on peut appliquer une tension électrique. Les trois broches portent les noms suivants: **Collecteur**, **Base** et **Émetteur**.

On distingue deux types de transistors bipolaires : les transistors NPN et les transistors PNP.



Transistor NPN.



Transistor PNP.

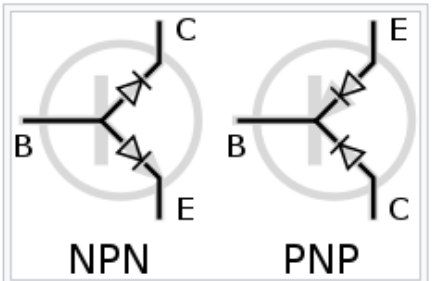
Il existe une tension entre chaque paire de broche, ainsi qu'un courant qui passe dans chaque broche. Cela fait en tout trois tensions notées V_{CE}, V_{CB}, V_{BE} et trois courants notés I_C, I_B et I_E.

Ceux-ci sont reliés par les équations suivantes :

$V_{CE} = V_{CB} + V_{BE}$

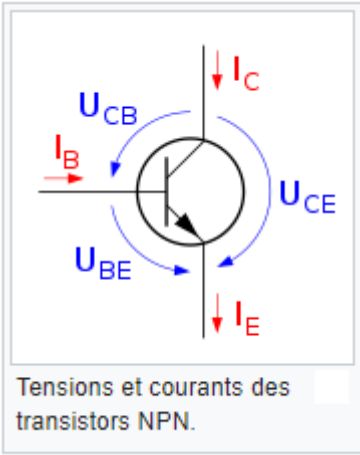
$I_E = I_B + I_C$

On peut considérer, à quelques détails près, qu'un transistor est composé de deux diodes mises en série dans des sens opposés :



Diodes internes d'un transistor de type BJT.

Le fonctionnement d'un transistor bipolaire NPN en interrupteur commandé consiste à activer la base, pour qu'elle permette au courant présent dans le collecteur de s'écouler jusqu'à l'émetteur.

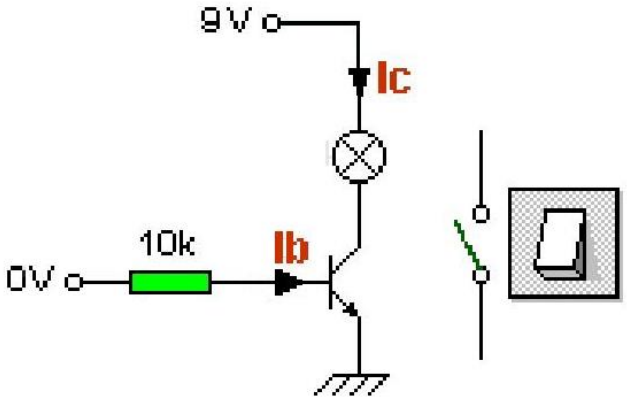


Tensions et courants des transistors NPN.

Quand le courant de base est nul, le transistor est bloqué :

$V_{BE} = 0 \rightarrow Ic = I_E = 0$

Il est équivalent à un interrupteur ouvert :

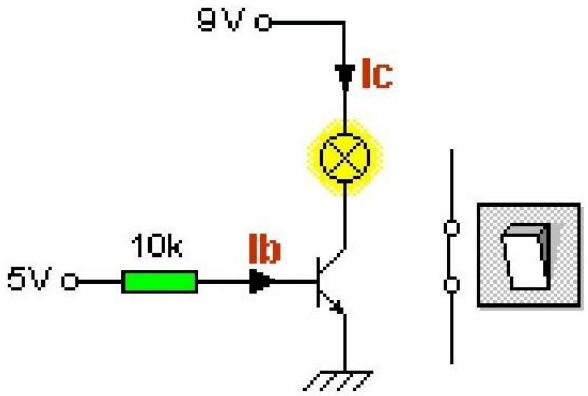


Quand le courant de la base est suffisant, le transistor est saturé :

V_{BE} = 0,7 V (tension de seuil de la diode base-émetteur), le transistor est alors passant.

Pour être saturé, il faut que : $\mathbf{I_b > I_c/\beta}$ où β est le gain en courant du transistor ($\mathbf{I_b \times \beta = Ic}$), aussi souvent appelé Hfe dans les fiches techniques des constructeurs.

Le transistor est alors équivalent à un interrupteur fermé :



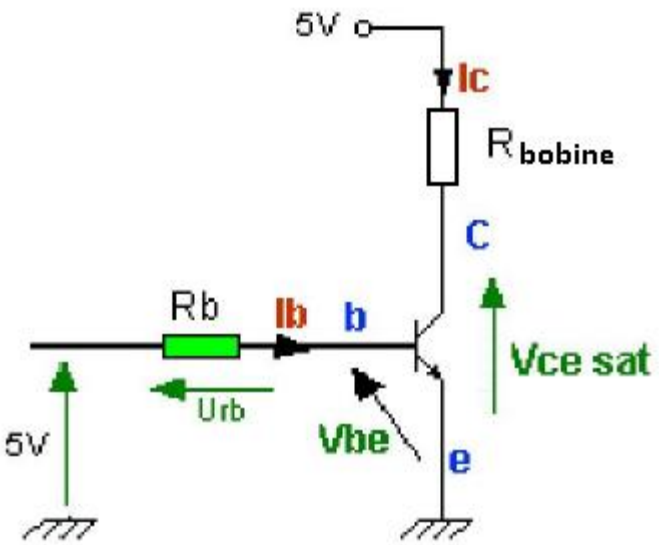
La résistance de la base doit être calculée pour avoir un courant I_b suffisant.

Avec le circuit de contrôle pour notre relais, on peut calculer la résistance de la base. En premier il faut déterminer le courant I_c :

$$\mathbf{I_c = U_{bobine\ relais} / R_{bobine\ relais}}$$

R_{bobine} = 120 Ω
U_{bobine} = 5 - $V_{ce\ sat}$

La tension $V_{ce\ sat}$ est proche de 0 V mais pas nulle. $V_{ce\ sat} \approx$ 0,2 V

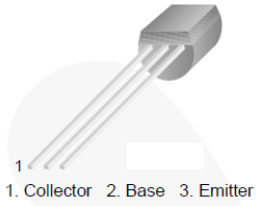


Donc : $\mathbf{I_c = 4,8 / 120 = 0,04\ A = 40\ mA}$

Le courant de la base I_b doit être suffisant pour saturer le transistor:

$\mathbf{I_b > I_c / ß}$

D'après la documentation du constructeur du transistor **BC547B**, ß est au moins égal à 200 :



h_{FE} Classification

Classification	A	B	C
h_{FE}	110 ~ 220	200 ~ 450	420 ~ 800

Il faut donc : $\mathbf{I_b\ min = 40 / 200 = 0,2\ mA}$

Connaissant I_b, il est maintenant possible de calculer Rb :

$R_b = U_{Rb} / I_b$

$V_{BE} + U_{Rb} = 5$ V avec $V_{BE} = 0.7$ V (tension de seuil de la diode)

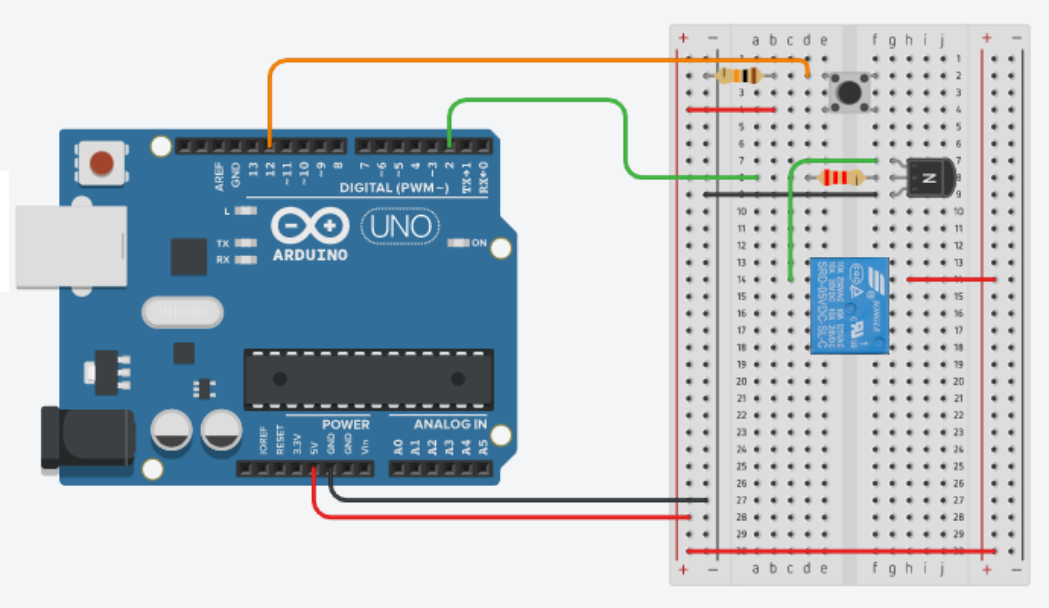
Donc : $U_{Rb} = 5 - Vbe = 5 - 0,7 = 4,3$ V

$Rb = 4,3 / 0,2.10^{-3} = 21\,500\ \Omega = 21,5\ k\Omega$

La résistance de la base doit donc être au maximum égale à 21,5 kΩ pour que le courant i_b soit au minimum de 0,2mA.

Dans notre circuit de contrôle du relais, la résistance R_b utilisée étant de 2,2 kΩ, le courant i_b est alors d'environ 2 mA.

Nous sommes donc assurés de saturer le transistor quand une sortie digitale de l'Arduino reliée à R_b est à un niveau haut (5V) et ceci sans danger pour la sortie de l'Arduino (rappel : I max sortie = 20 mA), selon le circuit suivant :



Avec le même programme que pour le contrôle du relais Grove, il est possible de contrôler le relais SRS-06VDC-SL :

- bouton appuyé :
  - → Sortie digitale 2 à HIGH
  - → Le transistor est saturé
  - → Un courant circule dans la bobine du relais
  - → Le relais commute en position NO

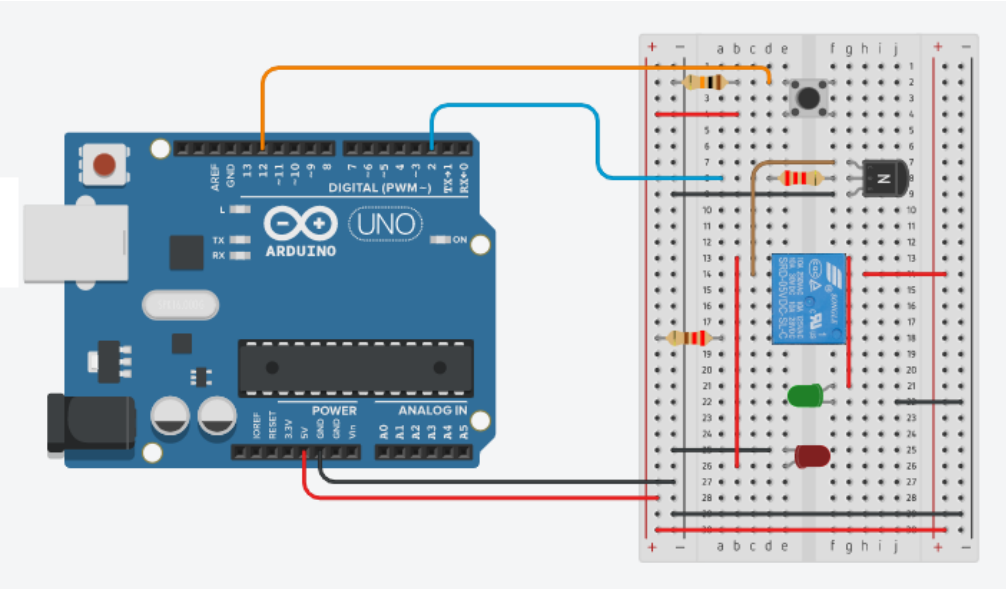
- bouton relâché :
  - → Sortie digitale 2 à LOW
  - → Le transistor est bloqué
  - → Aucun courant dans la bobine du relais
  - → Le relais commute en position NC

<u>Exemple d'application</u> :

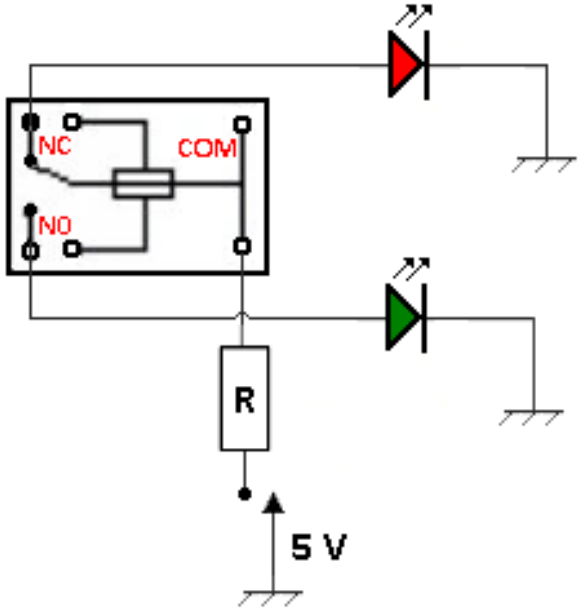
Le circuit suivant permet d'allumer la DEL rouge ou la DEL verte avec un bouton poussoir et le programme de contrôle d'un relais (**"Relais.ino" dans le dossier "Codes/Circuit RC"**) :

- bouton relâché : La DEL rouge est allumée et la DEL verte est éteinte,

- bouton appuyé : La DEL verte est allumée et la DEL rouge est éteinte.

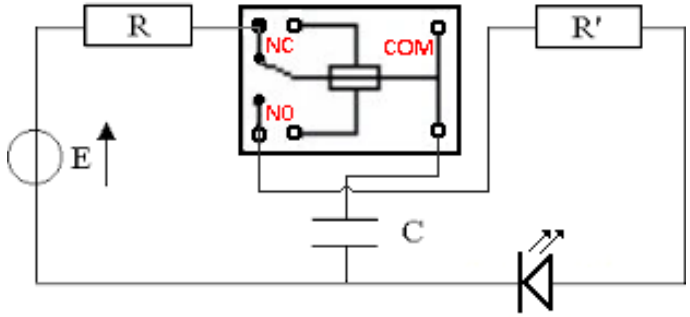


Le circuit de puissance est alors (relais vu de dessous) :



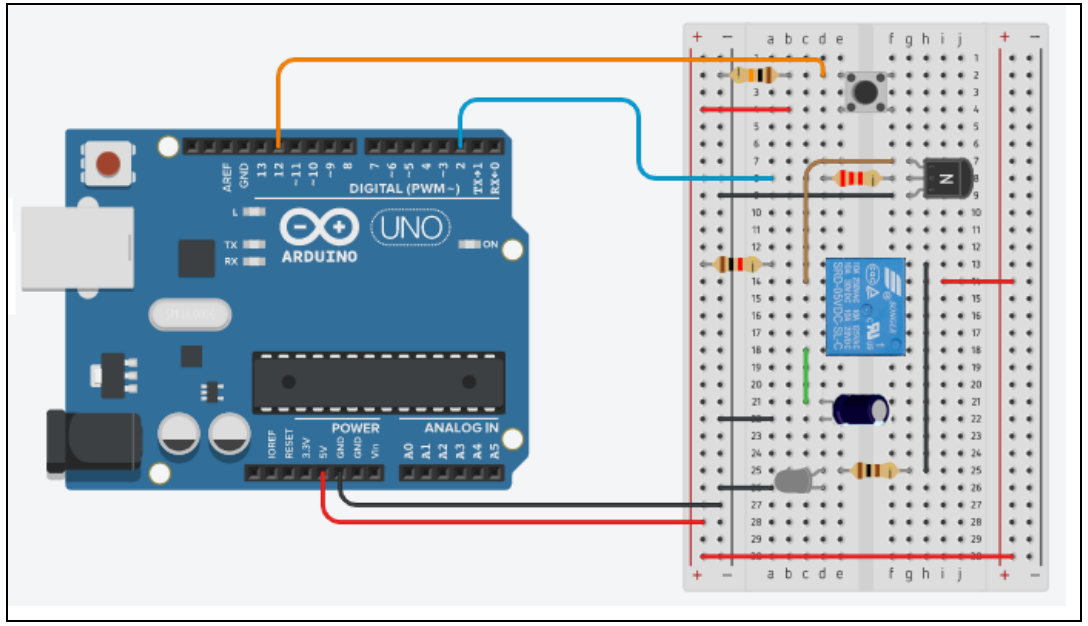
Avec R = 220 Ω

En transposant cet exemple à notre circuit de simulation de flash, le circuit de puissance devient :



Ainsi en l'absence de courant dans la bobine de l'électroaimant, le condensateur se charge à travers la résistance R et quand un courant circule dans la bobine, le condensateur se décharge dans R' et la DEL.

Le circuit de l'activité est alors :



. Liste des composants :

. 1 condensateur de 470 µF (C chimique : **attention à la polarité**)
. 1 résistance de 10 kΩ (résistance du bouton poussoir)
. 1 bouton poussoir
. 1 transistor bipolaire NPN (BC547B)
. 1 résistance de 2,2 kΩ (résistance du transistor)
. 1 relais SRS-06VDC-SL
. 1 résistance de 1 kΩ (résistance de charge du condensateur)
. 1 résistance de 100 Ω (résistance de décharge du condensateur)
. 1 DEL blanche

. Le programme

Voici le code de l'activité (**"Activity4.ino" dans le dossier "Codes/Circuit RC"**) :

Activity4

```
// Déclaration des constantes et variables

const int PinUC = 0;
const int PinButton = 12;
const int PinRelay = 2;

int ValPinUC = 0;
float UC = 0.0;
int StateHigh = 0;
int StateLow = 0;

int ValButton = 0;

// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton, INPUT);
  pinMode(PinRelay, OUTPUT);
  Serial.println("Charge du condensateur en cours.");
  digitalWrite(PinRelay, 0);
  while (ValPinUC<1022) {
    ValPinUC = analogRead(PinUC);
    }
  Serial.println("Condensateur charge.");
  Serial.print("Maintenez le bouton poussoir appuye ");
  Serial.println("pour decharger le condensateur.");
}

// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);

if (ValButton == HIGH)
  {
  if (StateHigh==0){
    Serial.println("Decharge du condensateur:");
    StateHigh = 1;
    StateLow = 0;
  }
  digitalWrite(PinRelay, 1);
  ValPinUC = analogRead(PinUC);
  UC = (ValPinUC/1023.0)*5.0;
  Serial.print("UC (V): ");
  Serial.println(UC,2);
  delay(200);
  }
```

```
else {
  if (StateLow==0){
    ValPinUC = analogRead(PinUC);
    if (ValPinUC<1022){
      Serial.println("Charge du condensateur:");
    }
    StateLow = 1;
    StateHigh = 0;
  }
  digitalWrite(PinRelay, 0);
  while (ValPinUC<1022) {
    ValPinUC = analogRead(PinUC);
    UC = (ValPinUC/1023.0)*5.0;
    Serial.print("UC (V): ");
    Serial.println(UC,2);
    delay(200);
    }
  }
}
```

Déroulement du programme :

- 1. Déclaration des constantes et variables :

. **const int PinUC = 0** (broche A0 du condensateur)
. **const int buttonPin = 12** (broche du bouton poussoir)
. **const int PinRelay = 2** (broche du relais)

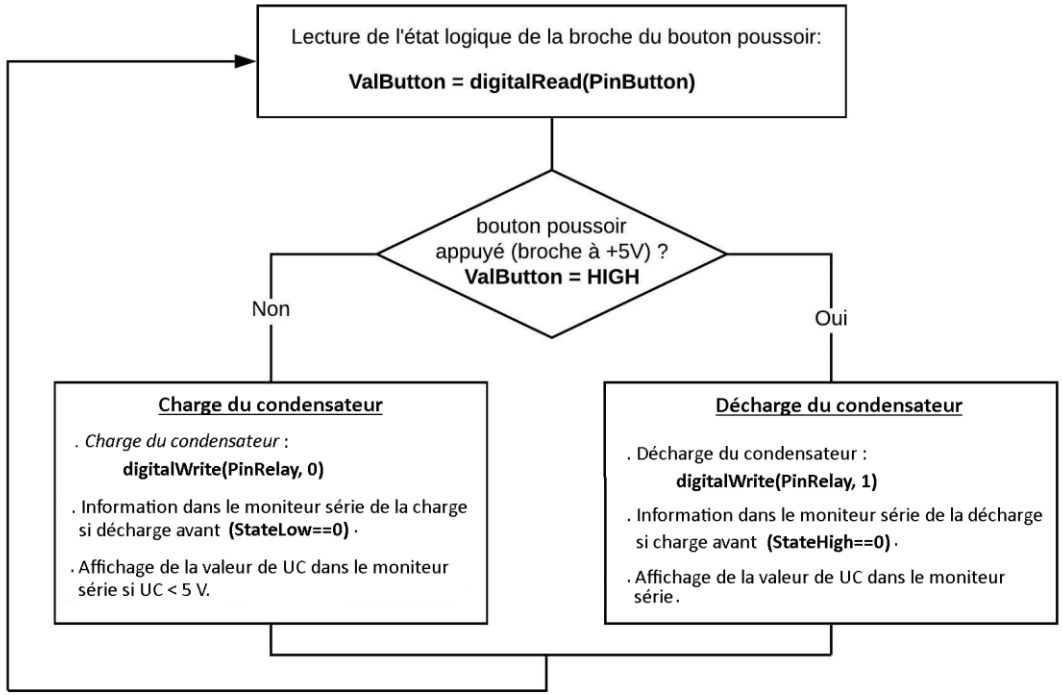
. **int ValPinUc = 0** (variable nombre entier valeur broche du condensateur)
. **float Uc = 0.0** (variable nombre décimal calcul tension Uc)
. **int StateHigh = 0** (variable nombre entier indiquant si état haut broche relais)
. **int StateLow = 0** (variable nombre entier indiquant si état bas broche relais)

. **int ValButton = 0** (variable nombre entier valeur broche bouton poussoir)

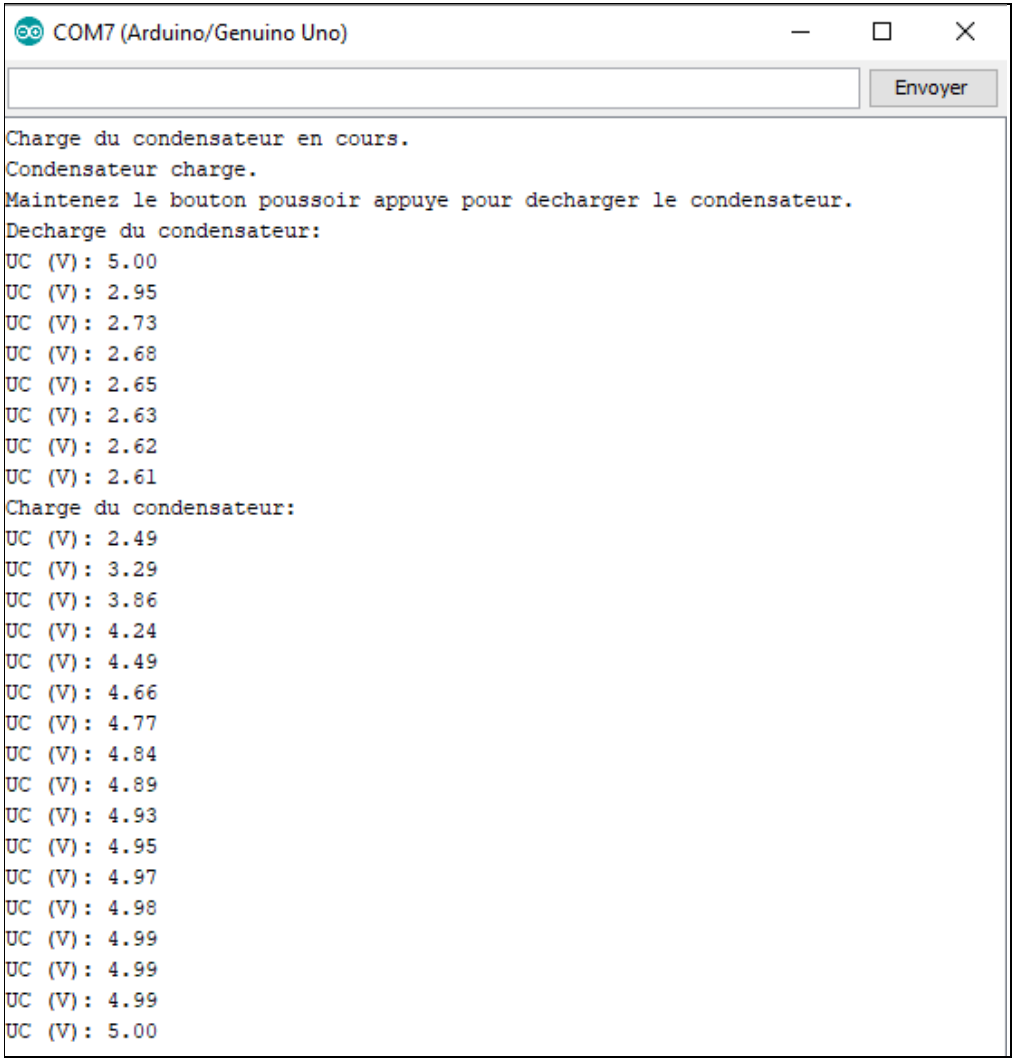
- 2. Initialisation des entrées et sorties :

. Initialisation de la broche du relais en sortie,
. Initialisation de la broche du bouton poussoir en entrée,
. Charge du condensateur.

- 3. Fonction principale en boucle :



Résultats dans le moniteur série :



Remarque :

On peut voir, dans le moniteur série, que le condensateur se décharge jusqu'à atteindre la tension de seuil de la DEL blanche (environ 2,5 V).
En effet, quand la tension aux bornes de la DEL est inférieure à la tension de seuil, la diode n'est plus passante, l'intensité dans le circuit est nulle et le condensateur ne se décharge plus.

Activité 5 : Réaliser un flash périodique

Dans cette activité, nous allons modifier le programme de l'activité précédente (circuit identique) pour que le flash s'effectue périodiquement après avoir appuyé sur le bouton poussoir.

Pour cela, il suffit de surveiller, en boucle, la tension Uc aux bornes du condensateur de façon à alterner entre la charge et la décharge :

- Si Uc est supérieure à une valeur **Uc max** à définir, le condensateur est déchargé, ce qui produit un flash.
- Si Uc est inférieure à une valeur **Uc min** à définir, le condensateur est chargé jusqu'à **Uc max**, puis le condensateur est déchargé jusqu'à Uc min et ainsi de suite…

Le flash périodique est arrêté en appuyant de nouveau sur le bouton poussoir.

. Le programme

Voici le code de l'activité (**"Activity5.ino" dans le dossier "Codes/Circuit RC"**), dont il sera possible de modifier les constantes **UcMax** et **UcMin** pour voir l'influence sur la période des flashs :

Activity5

```
// Déclaration des constantes et variables

const int PinUC = 0;
const int PinButton = 12;
const int PinRelay = 2;
const int UcMax = 1000;
const int UcMin = 800;

int ValPinUC = 0;
int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;



// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton, INPUT);
  pinMode(PinRelay, OUTPUT);
  digitalWrite(PinRelay, 0);
  while (ValPinUC<UcMax) {
    ValPinUC = analogRead(PinUC);
    }
  Serial.println("Appuyez sur le bouton poussoir pour commencer le flash.");
}


// Fonction principale en boucle

void loop() {

ValButton = digitalRead(PinButton);
delay(10);
```

```
if ((ValButton == HIGH)&&(OldValButton == LOW))
  {
  State=1-State;
  }
OldValButton = ValButton;

if (State==1)
  {
  if (OldState == 0)
    {
    Serial.println("Flash periodique en cours.");
    OldState=1;
    }
  ValPinUC = analogRead(PinUC);
  if (ValPinUC > UcMax)
  {
    digitalWrite(PinRelay, 1);
    while (ValPinUC>UcMin) {
      ValPinUC = analogRead(PinUC);
    }
  }
  else {
    digitalWrite(PinRelay, 0);
    while (ValPinUC<UcMax) {
      ValPinUC = analogRead(PinUC);
      }
    }
  }
else{
  if (OldState == 1){
    Serial.println("Fin du flash.");
    OldState = 0;
    digitalWrite(PinRelay, 1);
    ValPinUC = analogRead(PinUC);
    while (ValPinUC>UcMin) {
      ValPinUC = analogRead(PinUC);
    }
  }
}
}
```

Déroulement du programme :

- 1. Déclaration des constantes et variables :

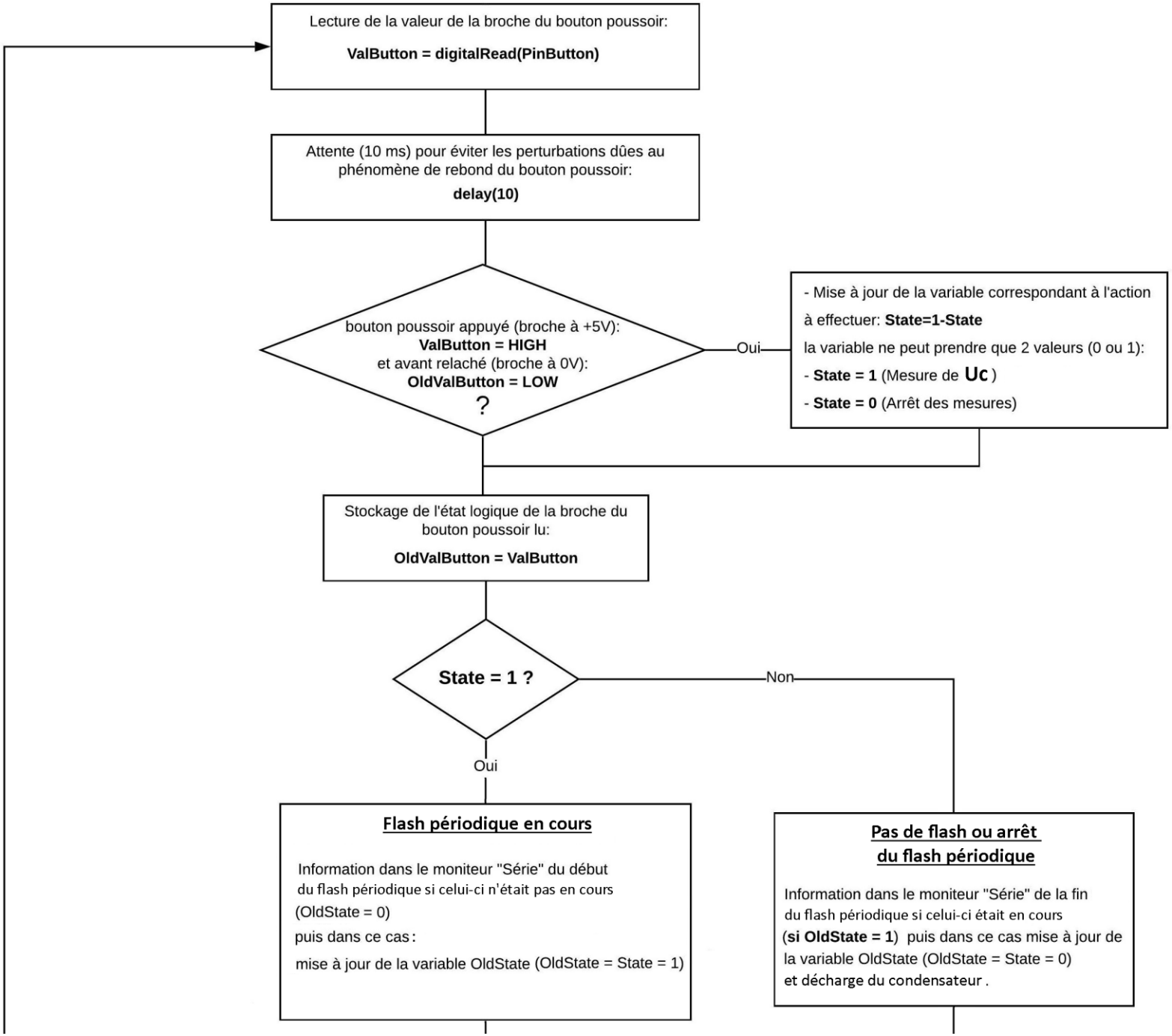
. **const int PinUc = 0** (broche du condensateur : A0)
. const int PinButton = 12 (broche du bouton poussoir)
. const int PinRelay = 2 (broche du relais)
. const int UcMax = 1000 (valeur maximale de la broche du condensateur)
. const int UcMin = 800 (valeur minimale de la broche du condensateur)

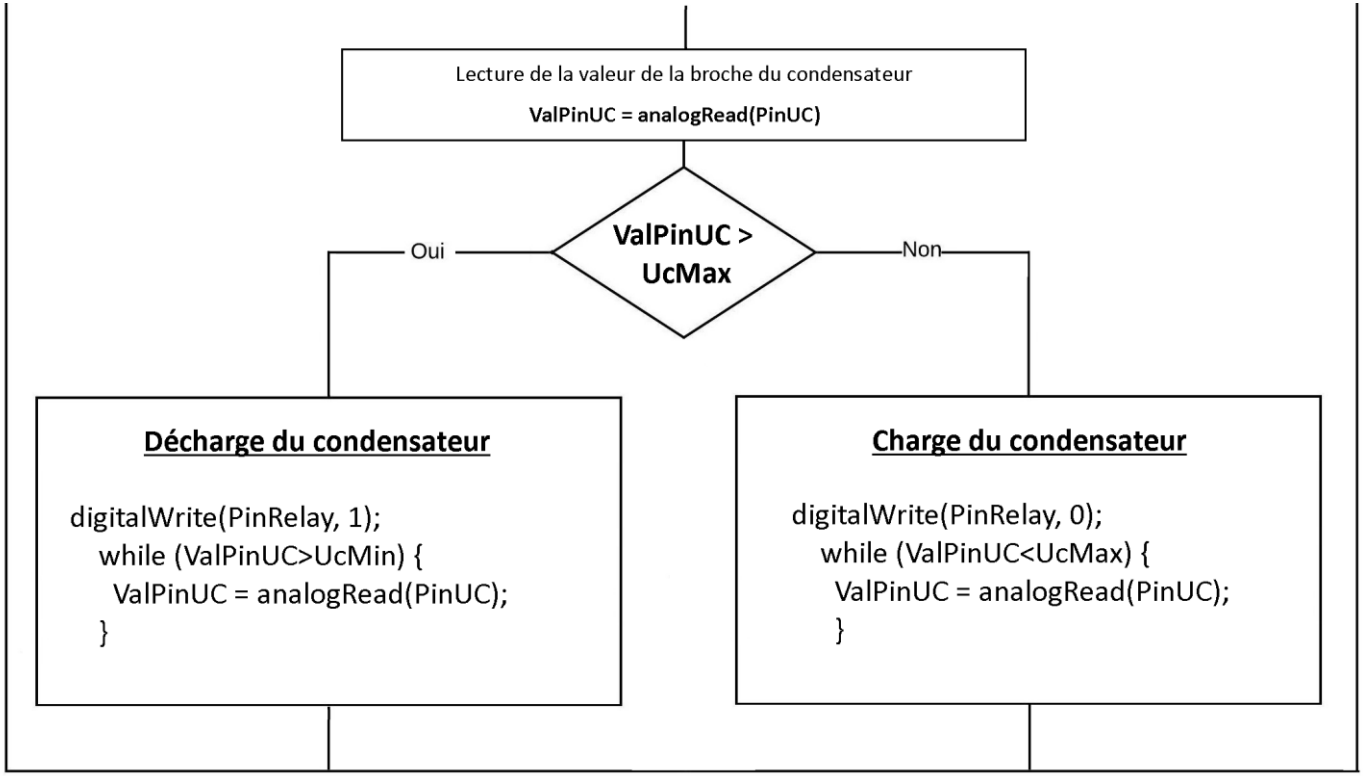
. **int ValPinUc = 0** (variable nombre entier valeur broche du condensateur)
. int ValButton = 0 (variable nombre entier valeur broche bouton poussoir)
. int OldValButton = 0 (variable nbr entier ancienne valeur broche btn poussoir)
. int State = 0 (variable nombre entier pour action à effectuer)
. int OldState = 0 (variable nombre entier pour action effectuée précédemment)

- 2. Initialisation des entrées et sorties :

. Initialisation de la broche du relais en sortie,
. Initialisation de la broche du bouton poussoir en entrée,
. Charge du condensateur jusqu'à UcMax.

- 3. Fonction principale en boucle :


Lecture de la valeur de la broche du bouton poussoir:
ValButton = digitalRead(PinButton)
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
delay(10)
bouton poussoir appuyé (broche à +5V):
ValButton = HIGH
et avant relaché (broche à 0V):
OldValButton = LOW
?
Oui
- Mise à jour de la variable correspondant à l'action à effectuer: State=1-State
la variable ne peut prendre que 2 valeurs (0 ou 1):
- State = 1 (Mesure de Uc)
- State = 0 (Arrêt des mesures)
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Flash périodique en cours
Information dans le moniteur "Série" du début du flash périodique si celui-ci n'était pas en cours (OldState = 0)
puis dans ce cas :
mise à jour de la variable OldState (OldState = State = 1)
Pas de flash ou arrêt du flash périodique
Information dans le moniteur "Série" de la fin du flash périodique si celui-ci était en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0)
et décharge du condensateur .



Résultats dans le moniteur série :