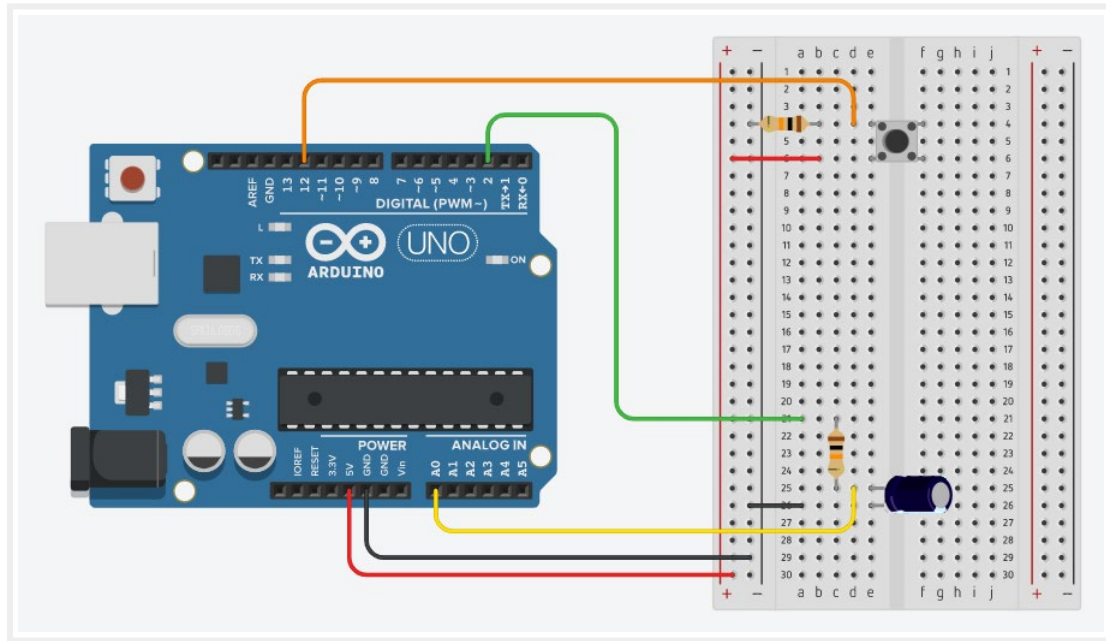


Dipôles RC – Constante de temps

(Détermination de la capacité d'un condensateur par mesure d'une constante de temps)



. Liste des composants :

- . 1 condensateur de 100 μF (C chimique : **attention à la polarité**)
- . 2 résistances de 10 $\text{k}\Omega$ (résistance du circuit du bouton poussoir et du dipôle RC)
- . 1 bouton poussoir
- . 1 plaque d'essais
- . Fils de connexion

. Objectif

Dans cette activité, nous allons modifier le programme de l'activité "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 $U_c=f(t)$.

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

$$\tau = RC \quad \text{donc :} \quad C = \tau / R$$

. 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 $t = \tau$: $U_c = 0,63 \times 5 = 3,15$ V).

Avec le même circuit que les activités de charge et décharge d'un condensateur, 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 : **$0,63 \times 1023 = 644$** (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 :

```
Dipoles_RC_Tau
// 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 : 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 pour stocker la valeur de la broche du condensateur)
- . **unsigned long t0** (variable nombre entier long pour stocker la date du début de charge)

- . **float dt** (variable nombre décimal pour stocker la différence de temps entre les mesures de Uc)
- . **int ValButton = 0** (variable nombre entier pour stocker la valeur de la broche du bouton poussoir)
- . **int OldValButton = 0** (variable nombre entier pour stocker la valeur précédente de la broche du bouton poussoir)
- . **int State = 0** (variable nombre entier correspondant à l'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)

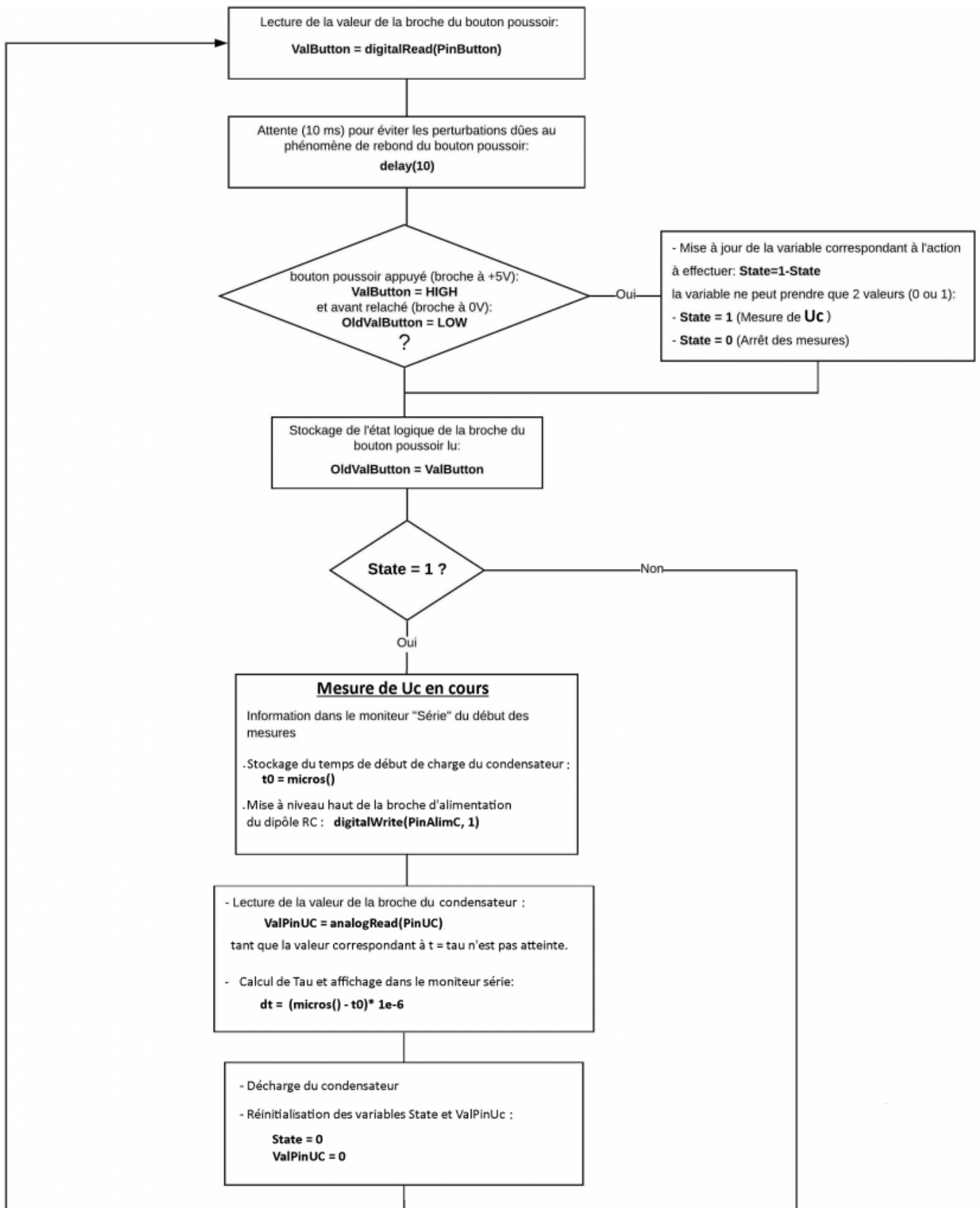
– Attente de la fin de la décharge du condensateur :

while (analogRead(PinUC) > 0))

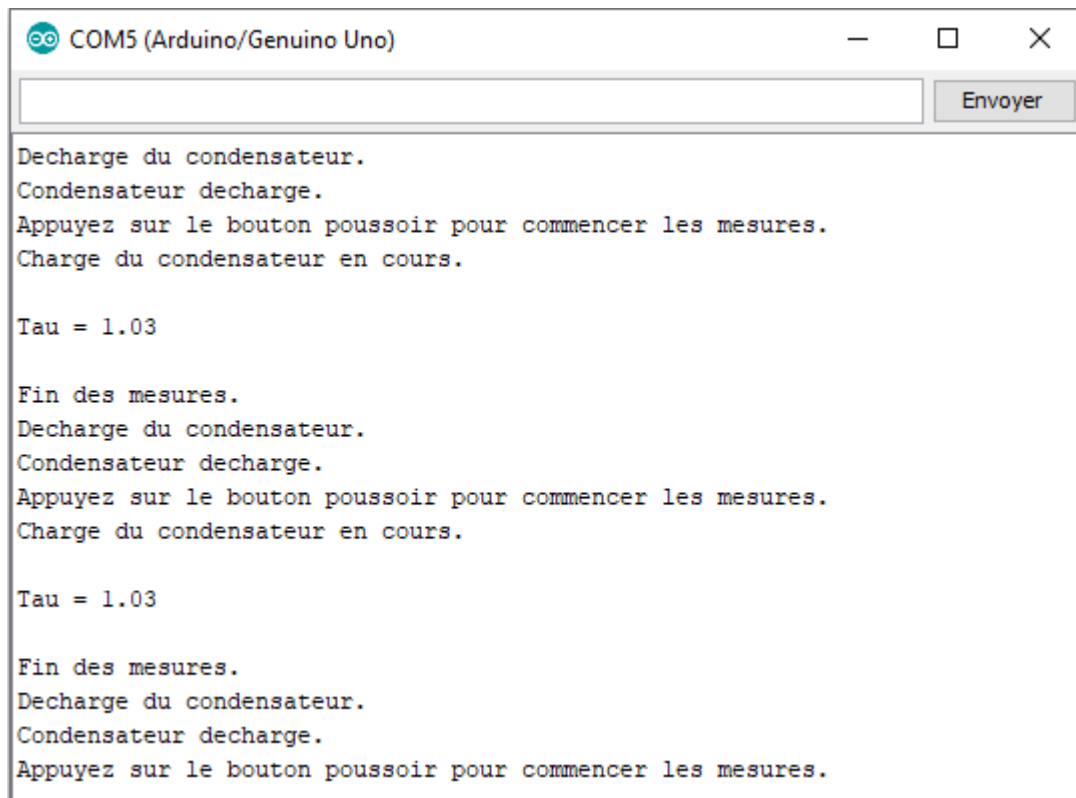
– 3. Initialisation des entrées et sorties :

- . **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 :



Résultats dans le moniteur série



```
COM5 (Arduino/Genuino Uno)

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

Tau = 1.03

Fin des mesures.
Decharge du condensateur.
Condensateur decharge.
Appuyez sur le bouton poussoir pour commencer les mesures.
Charge du condensateur en cours.

Tau = 1.03

Fin des mesures.
Decharge du condensateur.
Condensateur decharge.
Appuyez sur le bouton poussoir pour commencer les mesures.
```

. [Exploitation des mesures](#)

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

On peut alors déterminer la capacité du condensateur (valeur théorique = 100 μF) :

$$C = \tau / R = 1,03 / 10 \cdot 10^3 = 1,03 \cdot 10^{-4} \text{ F} = 103 \mu\text{F}$$