TP Détecteur de passager

Capacités exigibles du programme :

— Décrire qualitativement le phénomène d'influence électrostatique.

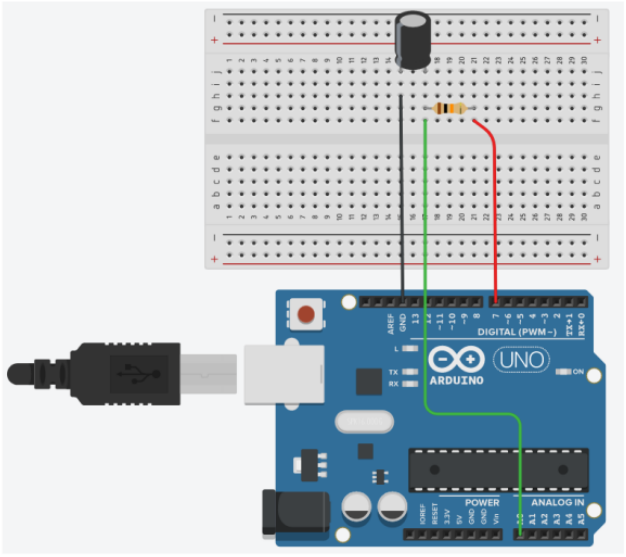
Liste du matériel :

— Plaquette de montage de composants
— Résistances de 10 kΩ, 1 (ou 10) MΩ
— Condensateur de 100 μF
— Condensateur artisanal (2 feuilles d'aluminium plastifiées, chacune connectée à un petit fil)
— Module Arduino, câble USB
— GBF
— Oscilloscope
— Masses marquées assez lourdes (un jeu pour toute la classe)

1 Suivi de la charge d'un condensateur avec un module Arduino

Suivi de tension à base d'Arduino :
Déterminer et mettre en œuvre un protocole expérimental permettant, à l'aide d'un module Arduino, de récupérer l'évolution de la tension aux bornes d'un condensateur en fonction du temps lors de sa charge.

Représenter graphiquement le résultat sur un notebook Python.



On prendra $R = 10$ kΩ et $C = 100$ μF.

2 Mesure de la constante de temps d'un circuit RC

Mesure de la constante de temps :
Déterminer et mettre en œuvre un protocole expérimental permettant, à l'aide d'un module Arduino, de mesurer la constante de temps d'un circuit RC lors de la charge d'un condensateur.

On prendra $R = 10$ kΩ et $C = 100$ μF. On pourra s'inspirer du programme Arduino préparé pour l'activité suivante.

3 Élaboration d'un détecteur de passager

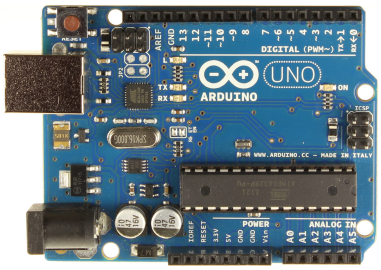
Élaboration d'un détecteur de passager :
Déterminer et mettre en œuvre un protocole expérimental permettant, à l'aide d'un module Arduino, de réaliser un détecteur de passager, comme ceux que l'on peut trouver sur les sièges avant d'un véhicule.

On réalisera enfin un étalonnage grossier du capteur permettant de mesurer approximativement la masse présente sur le siège passager.

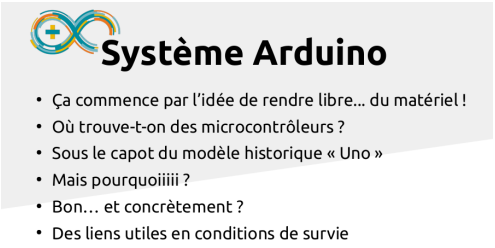
A Annexe 1 - Principe d'un suivi de tension à l'aide d'un module Arduino

A.1 Principe

Le principe est d'utiliser un module Arduino, possédant un microcontrôleur programmable pouvant interagir avec un ensemnle de broches E/S, pour récupérer une tension au cours du temps.



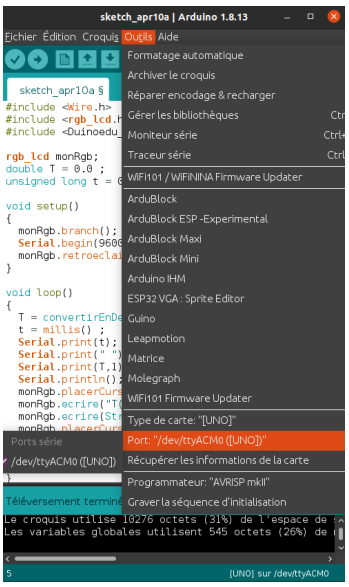
Un diaporama de présentation permet d'introduire l'histoire de ce module :



Ainsi, voici quelques principes fondamentaux sur le module Arduino UNO :

- le microcontrôleur peut interagir avec les différentes broches E/S ;
- chaque broche E/S fonctionne entre 0 et 5 V (**pas de tension négative**) ;
- toutes les tensions sont relatives à la masse `GND` correspondant à 0 V ;
- certaines broches (analogiques) permettent de **mesurer** une tension **entre** 0 et 5 V (broches `A0` à `A5`) ;
- d'autres broches (numériques) permettent de **mesurer ou délivrer** une tension de 0 **OU** 5 V (broches `0` à `13`) ;
- un IDE permet d'écrire le programme en langage C++, qu'il faut alors compiler et téléverser vers le module Arduino.

Remarque : Juste après avoir connecté le module Arduino à l'ordinateur, il est indispensable de choisir le bon port de communication :



A.2 Mesurer une tension sur une broche analogique

Prenons l'exemple de la mesure d'une tension sur l'entrée analogique `A0`. La fonction à utiliser dans le programme est `analogRead(A0)`. Cette fonction retourne une valeur N entre 0 et 1023, qui correspond à l'échelle 0-5 V. Ainsi la tension est alors :

$$u = \frac{N}{1023} \times 5$$

Un programme pour mesurer et afficher cette tension pourrait ainsi être :

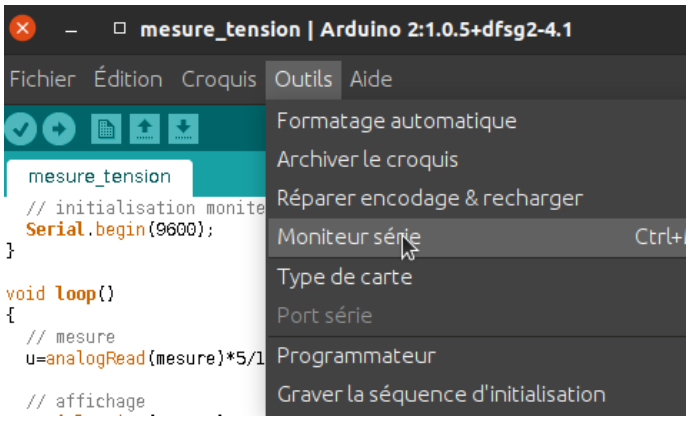
```
// broches
const int(mesure)=A0;

// grandeurs
float u;

void setup()
{
  // initialisation moniteur serie
  Serial.begin(9600);
}

void loop()
{
  // mesure
  u=analogRead(mesure)*5.0/1023;

  // affichage
  Serial.print("u : ");
  Serial.print(u);
  Serial.println(" V");

  delay(1000); // attente de 1 s
}
```

mesure_tension/mesure_tension.ino

Plusieurs commentaires s'imposent :

- contrairement au langage Python, l'indentation n'a aucune fonction en C++ (si ce n'est faciliter la lecture), les blocs de code sont définis par des accolades `{}` ;
- contrairement au langage Python, les grandeurs utilisées doivent être définies avec leur type en début de programme ;
- le code est constitué de deux fonctions (`void`), une première `setup()` ne sera exécutée qu'une seule fois au lancement, une seconde `loop()` qui sera exécutée en boucle (en permanence) ;
- les commentaires sont introduits par `//`.

L'affichage du résultat de la mesure se fait à l'aide des fonctions `Serial.print()` (afficher) et `Serial.println()` (afficher puis aller à la ligne). Les données à afficher sont envoyées sur le port série pour être affichées dans le **moniteur série**, accessible **pendant l'exécution des instructions sur le module Arduino** via le menu *Outils* → *Moniteur série*.



La vitesse de communication est définie par la fonction `Serial.begin()` et s'exprime en bauds (nombre d'éléments transmis par seconde).

On peut remarquer l'utilisation de la fonction `delay()` pour marquer une pause, ici de 1000 ms.

À l'aide d'un fil, reliez la broche `3.3V` à la broche `A0`. Téléverser le programme précédent sur le module puis lancer le moniteur série.

Ainsi, on met la broche `A0` au potentiel 3,3 V, la mesure donnera donc comme résultat $u = V_{A0} - V_{GND} = 3,3 - 0 = 3,3$ V. L'affichage du moniteur série doit donner le résultat ci-contre.

A.3 Mesurer un temps

Une horloge est nécessairement présente dans le microcontrôleur. On peut par exemple accéder au temps écoulé depuis l'initialisation de la carte (par exemple lors du téléversement du programme), à l'aide de la fonction `millis()` qui retourne ce temps en ms.

Modifier le programme précédent pour que le moniteur série affiche, en plus de la tension u, le temps t écoulé depuis l'initialisation de la carte.

A.4 Imposer une tension sur une broche numérique

On peut vouloir imposer une tension de 0 V ou 5 V sur une broche numérique (on ne peut pas imposer autre chose), cela se fait de la façon suivante :

- on définit la broche numérique dans la fonction `setup()` comme une broche de sortie avec `pinMode(numero_broche,OUTPUT)` ;
- on définit la tension à imposer avec `digitalWrite(numero_broche,HIGH)` pour imposer 5 V ou `digitalWrite(numero_broche,LOW)` pour imposer 0 V.

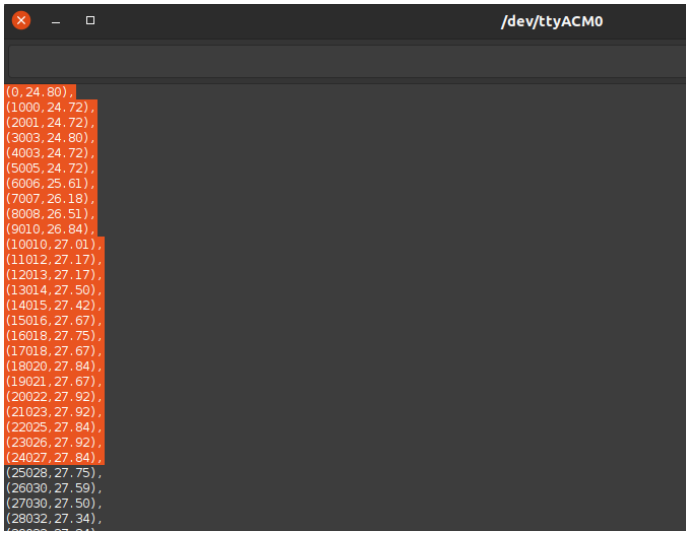
Modifier le programme précédent pour que le moniteur série affiche la tension u imposée par la broche numérique 2, basculant alternativement de 0 V à 5 V, toutes les secondes environ, ainsi que le temps t écoulé depuis l'initialisation de la carte.

On utilisera le type `unsigned long` pour t.

A.5 Mesure d'une tension au cours du temps et récupération des données

Si l'on doit récupérer les données pour en faire par exemple une représentation graphique en Python, le plus simple est de réaliser un copier-coller des données directement depuis le moniteur série.

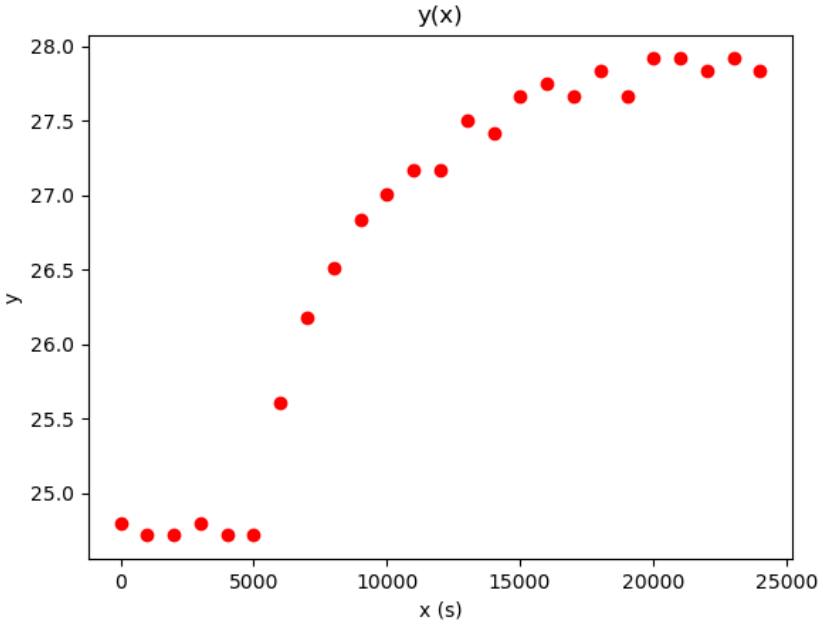
Il faut donc les afficher d'une façon qui permette de les incorporer facilement dans un programme Python. Une possibilité est de les afficher dans le moniteur série sous la forme `(mesure_x,mesure_y),` à chaque ligne, par exemple :



Puis d'utiliser un script Python où il suffit de coller les données sélectionnées pour réaliser leur représentation graphique, par exemple :

```python
import matplotlib.pyplot as plt

donnees=[
(0,24.80),
(1000,24.72),
(2001,24.72),
(3003,24.80),
(4003,24.72),
(5005,24.72),
(6006,25.61),
(7007,26.18),
(8008,26.51),
(9010,26.84),
(10010,27.01),
(11012,27.17),
(12013,27.17),
(13014,27.50),
(14015,27.42),
(15016,27.67),
(16018,27.75),
(17018,27.67),
(18020,27.84),
(19021,27.67),
(20022,27.92),
(21023,27.92),
(22025,27.84),
(23026,27.92),
(24027,27.84),
]

liste_x=[mesure[0] for mesure in donnees]
liste_y=[mesure[1] for mesure in donnees]

plt.figure(1)
plt.plot(liste_x,liste_y,'ro')
plt.xlabel('x (s)')
plt.ylabel('y')
plt.title('y(x)')
plt.show()
```

recuperer_donneees_arduino_moniteur_serie_copier–coller.py

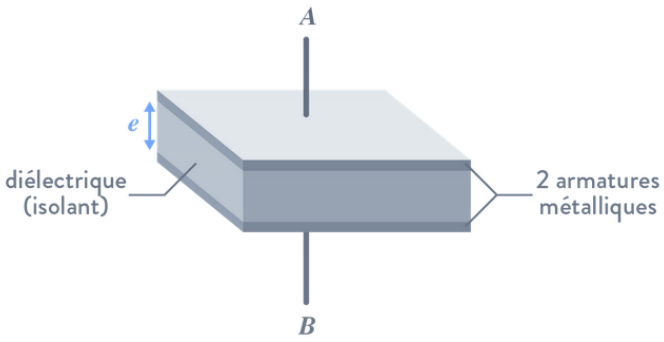


B Annexe 2 - Le condensateur

Un condensateur est un ensemble de deux conducteurs en influence totale. Pour un condensateur plan (cf. cours) nous avons établi que la capacité était donnée par :

$$C = \frac{\epsilon S}{e}$$

avec e l'épaisseur de l'isolant (diélectrique), S sa surface et $\epsilon = \epsilon_r \epsilon_0$ la permittivité diélectrique de l'isolant.



e : épaisseur

On donne les valeurs des permittivités diélectriques relatives ϵ_r pour différents matériaux :

air	1	paraffine	2.2	porcelaine	entre 5 et 6
bakélite	5	PVC	5	stéatite	5.8
caoutchouc	4	plexiglas	3.3	styroflex	2.5
caoutchouc siliconé	4.2	polyester	3.3	teflon	2.1
carton	4	polyéthylène	2.25	verre	entre 5 et 7
papier	2	polypropylène	2.2	stratifé-verre-époxy	5
papier bakélisé	5	polystyrène	2.4	mica	6