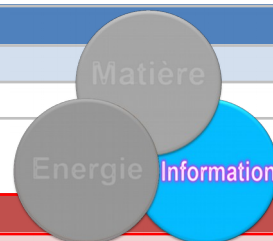


Terminale	Séquence N°1 / Thème N°2	
STI2D spécialité SIN	L'environnement de prototypage Arduino Révisions de 1ère (programmation, CAN, CNA, algo, ...)	

Pré-requis :

Connaissance des structures algorithmiques de l'informatique
Méthode, principes et caractéristiques de la programmation en C (Arduino)



Compétences visées :

Comprendre et mettre en œuvre des structures algorithmiques
Compréhension de la notion de variable informatique
Révisions sur le codage (binaire,)

Plan des 2 séances :

Partie N°3 : en bonus : d'autres exercices sur prototypes à base d'Arduino.....1



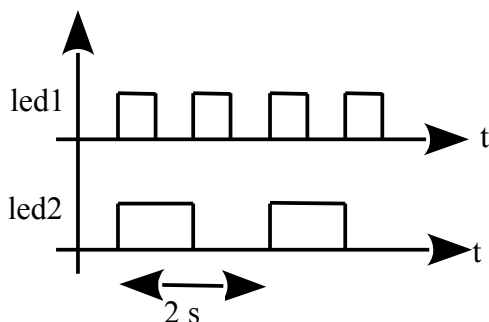
Une grande partie des informations utiles sont présentes dans les cours et TP de 1ère STI (I2D).

Partie N°3 : en bonus : d'autres exercices sur prototypes à base d'Arduino

Exercice 6 : clignotement de 2 leds (révision/découverte de la fonction millis)

Objectif : faire clignoter 2 led à 2 fréquences différentes : 1Hz et 0,5 Hz

Voici le chronogramme du fonctionnement que l'on veut obtenir:



Le programme proposé :

```
// définition des variables
const int nombre_Pins = 3; // .....
const int ledPin1 = 13; // the LED pins
const int ledPin2 = 12; // the LED pins
```

```

int interval1 = 500; // .....
int interval2 = 1000; // .....

void setup()
{
    pinMode(ledPin1, OUTPUT);
    pinMode(ledPin2, OUTPUT);
}

void loop()
{
    digitalWrite(ledPin1,HIGH);
    delay(interval1);
    digitalWrite(ledPin1,LOW);
    delay(interval1);
    digitalWrite(ledPin2,HIGH);
    delay(interval2);
    digitalWrite(ledPin2,LOW);
    delay(interval2);
}

```

1. *Faire l'algorithme (sous forme d'algorigrammes) du programme*
2. *Le programme ne donne pas le résultat espéré. Faire le chronogramme de l'état des leds que génère ce programme.*

On se propose maintenant d'essayer une autre version du programme:

Données techniques :

unsigned long : Unsigned long variables are extended size variables for number storage, and store 32 bits. Unlike standard longs unsigned longs won't store negative numbers, making their range from 0 to 4294967295

Millis()

Returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.

Exemple: variable (type unsigned long) = millis()

Programme :

```

// définition des variables
const int nombre_Pins = 3; // .....
const int ledPin1 = 13; // the LED pins
const int ledPin2 = 12; // the LED pins

int interval1 = 500; // .....
int interval2 = 1000; // .....
boolean etat_led1 = LOW;
boolean etat_led2 = LOW;
unsigned long temps1, temps2;

```

```
void setup()
```

```

{
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  temps1= millis();
  temps2= millis();
}

void loop()
{
  // gestion
  if (millis()-temps1>499)
  {
    temps1=millis();
    if (etat_led1==LOW)
    {
      etat_led1=HIGH;
    }
    else etat_led1=LOW;
  }

  if (millis()-temps2>999)
  {
    temps2=millis();
    if (etat_led2==LOW)
    {
      etat_led2=HIGH;
    }
    else etat_led2=LOW;
  }

  // actions
  if (etat_led1==LOW)
  {
    digitalWrite(ledPin1,LOW);
  }
  else digitalWrite(ledPin1,HIGH);

  if (etat_led2==LOW)
  {
    digitalWrite(ledPin2,LOW);
  }
  else digitalWrite(ledPin2,HIGH);
}

```

3. *Faire l'algorithme (sous forme d'algorithmes) du programme*

4. *Expliquez, en détail, le fonctionnement du programme puis complétez les cases vides du tableau suivant concernant la led1 :*

Temps donné par millis()	temps1	Millis()-temps1	Etat_led1	Action ou commentaire
0	0 (dans le setup)	0	0	Démarrage du prog
1	0	1	0	rien
.....	0		0	rien

499				rien
500				
	500			Mémorisation du temps
501				
502				
.....				
999				
1000				
1001				
1002				
....				

5. Le programme fonctionne-t-il correctement par rapport à ce qui était demandé ?

Exercice 7 : allumage de plusieurs leds et boutons poussoirs

Rappel technique : la durée d'un cycle Arduino (temps d'exécution du programme et de lecture/écriture des entrées/sorties) est beaucoup plus court que la durée d'appui sur un bouton poussoir (entre 0,5 et 2 secondes). Bref l'appui sur un BP est vu plusieurs fois par la boucle "LOOP" du programme.

Soit le programme Arduino suivant :

```
// déclaration des variables, broches (pin)
const int btn_moins = 2;
const int btn_plus = 3;
const int led_0 = 10;
const int led_1 = 11;
const int led_2 = 12;
const int led_3 = 13;

int nombre_led = 0; // le nombre qui sera incrémenté et décrémenté
int etat_bouton; //lecture de l'état des boutons (un seul à la fois donc une variable suffit)
int memoire_plus = HIGH; //état appuyé par défaut
int memoire_moins = HIGH;

void setup()
{
  pinMode(btn_plus, INPUT);
  pinMode(btn_moins, INPUT);
  pinMode(led_0, OUTPUT);
  pinMode(led_1, OUTPUT);
  pinMode(led_2, OUTPUT);
  pinMode(led_3, OUTPUT);
}

2_prototypage_arduino_eleves (bonus).odt
```

```

    pinMode(led_2, OUTPUT);
    pinMode(led_3, OUTPUT);
}

void loop()
{
    etat_bouton = digitalRead(btn_plus); // lecture de l'état du bouton d'incrémement etat_

    //Si le bouton a un état différent que celui enregistré ET que cet état est "appuyé"
    if ((etat_bouton != memoire_plus) && (etat_bouton == LOW))
    {
        nombre_led++; //on incrémente la variable qui indique combien de LED à allumer
    }

    memoire_plus = etat_bouton; //on enregistre l'état du bouton pour le tour suivant

    etat_bouton = digitalRead(btn_moins); //lecture de son état

    //Si le bouton a un état différent que celui enregistré ET que cet état est "appuyé"
    if ((etat_bouton != memoire_moins) && (etat_bouton == LOW))
    {
        nombre_led--; //on décrémente la valeur de nombre_led
    }
    memoire_moins = etat_bouton; //on enregistre l'état du bouton pour le tour suivant

    //on applique des limites au nombre pour ne pas dépasser 4 ou passez sous 0
    if(nombre_led > 4) {
        nombre_led = 4;
    }

    if(nombre_led < 0) {
        nombre_led = 0;
    }

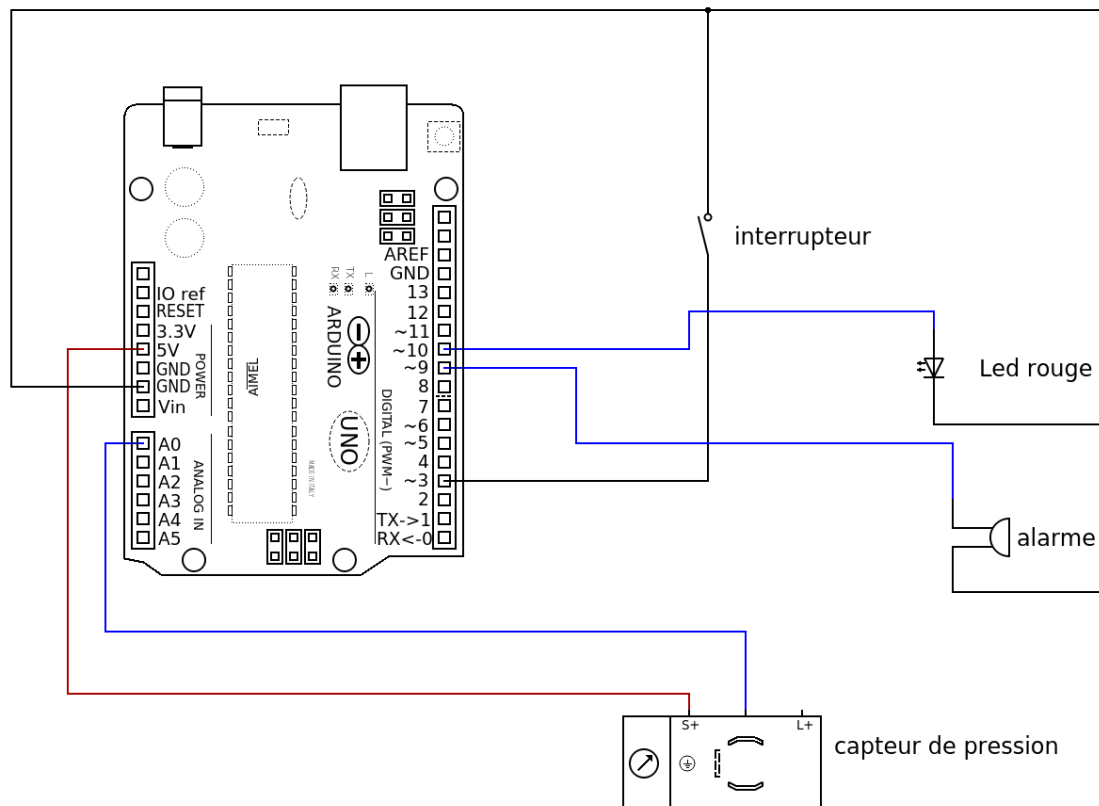
    //on éteint toutes les leds
    digitalWrite(led_0, LOW);
    digitalWrite(led_1, LOW);
    digitalWrite(led_2, LOW);
    digitalWrite(led_3, LOW);
}
//Puis on les allume une à une
if( nombre_led >= 1)
{
    digitalWrite(led_0, HIGH);
}
if( nombre_led >= 2)
{
    digitalWrite(led_1, HIGH);
}
if( nombre_led >= 3)
{
    digitalWrite(led_2, HIGH);
}
if( nombre_led >= 4)
{
    digitalWrite(led_3, HIGH);
}
} // fin du loop

```

1. *Faire l'algorithme (sous forme d'algorigrammes) du programme*
2. *Expliquez, en détail, ce que va faire le programme et comment il le fait.*

Exercice 8: réalisation du programme de gestion d'un capteur

Nous allons lire un capteur de pression à l'aide d'une carte Arduino. Celui-ci est branché sur une entrée de la carte Arduino (voir schéma).



Nom des variables à utiliser dans le programme :

- **capteur_press** (lecture du capteur)
- **press_volt** (conversion en volts de la valeur du capteur)
- **pression** (résultat du calcul en hPa de la pression mesurée)

Analyse de la documentation arduino :

1. *Quelles sont les valeurs possibles (plage de valeurs en volts) sur une entrée analogique de la carte Aruino Uno ?*
2. *Sur combien de bits fonctionne le convertisseur A/N des entrées analogiques ? Combien de valeurs différentes possibles pouvons-nous donc obtenir sur une entrée analogique ?*

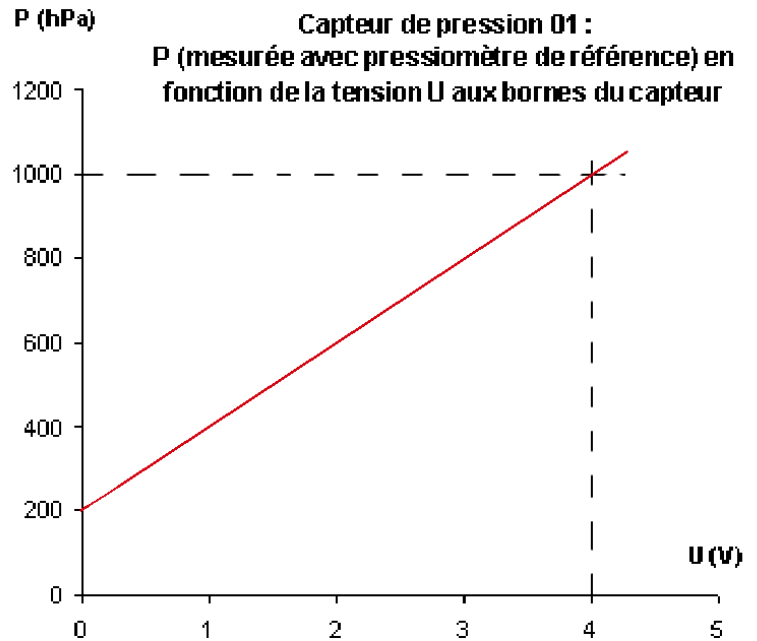
Application à notre capteur de pression :

3. *En vous aidant du câblage, faites le programme Arduino afin de lire le capteur de pression, seulement quand l'interrupteur en activé (la led sera allumée). Vous devez utiliser la variable "capteur_press" qui contiendra la lecture de l'entrée analogique (pensez à définir le type de cette variable).*
4. *Modifiez le programme pour que celui-ci calcule la valeur en volt (à partir de la lecture du capteur que vous avez mis dans la variable "capteur_press"). Vous mettrez votre résultat dans la variable "press_volt" (celle-ci permettra, à la question suivante, de calculer la pression mesurée à partir de cette valeur en volts).*

La courbe $P=f(U)$ caractéristique du capteur est la suivante :

On remarque qu'il s'agit d'une fonction affine (forme $y=ax+b$). Il est alors facile de trouver a et b à partir de 2 points tirés de la courbe (le point correspondant à $x=0$ et un point quelconque par exemple).

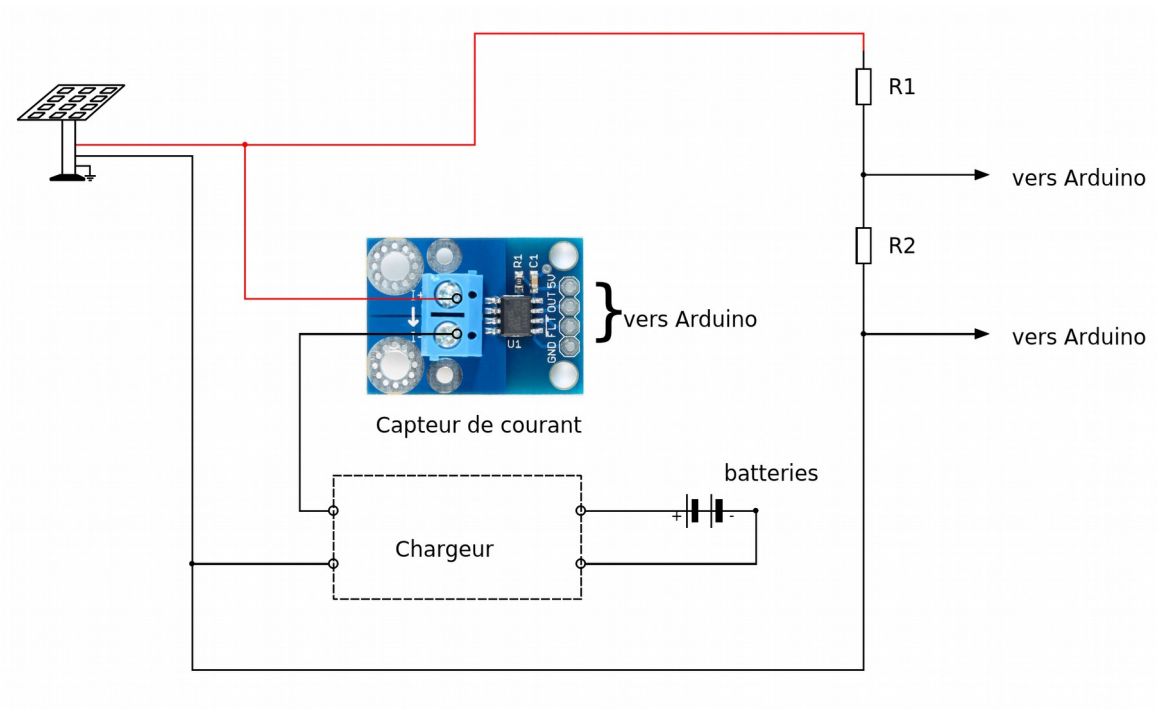
5. ***Vous allez maintenant modifier le programme Arduino pour que la variable "pression" contienne, après le calcul que vous devez tirer de la courbe précédente, la valeur de la pression en hPa.***
6. ***Vous allez maintenant modifier le programme arduino pour que le buzzer (alarme) soit activé quand la pression est supérieure à 850 hPa.***



Exercice 9: réalisation du programme de gestion d'un capteur de courant et lecture d'une tension

L'objectif de notre réalisation est de récupérer la tension et le courant fourni par un panneau solaire photovoltaïque (constitué de 2 cellules de 12V câblées en série).

Le schéma sera le suivant :



Nom des variables à utiliser dans le programme :

- **capteur_courant** (lecture du capteur)
- **courant_volt** (conversion en volts de la valeur du capteur)
- **courant** (résultat du calcul en Ampère du courant électrique mesuré)
- **tension** (valeur de la tension)

Partie 1 : gestion du capteur de courant

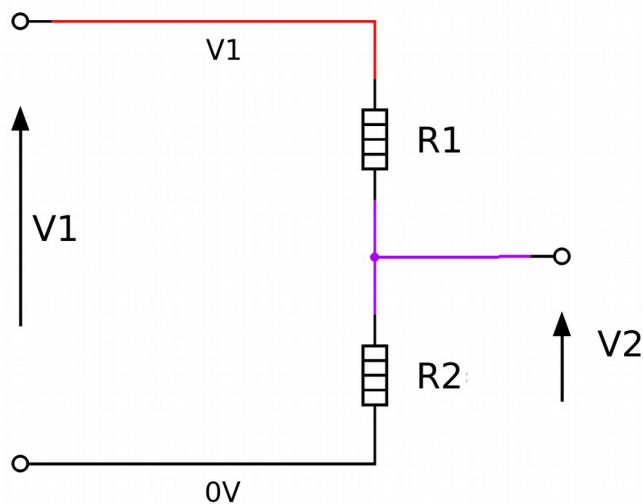
Données techniques :

- capteur linéaire
- alimentation $5V_{CC}$
- sortie : $0-5V_{CC}$
- plage de mesure: -12A à 12A

1. *Quelle sera la valeur fournie par le capteur lorsque le courant vaudra -12A ?*
2. *Quelle sera la valeur fournie par le capteur lorsque le courant vaudra 12A ?*
3. *Quelle sera la valeur fournie par le capteur lorsque le courant vaudra 0A ?*
4. *Sachant que le capteur sera câblé sur l'entrée analogique A1, faites le programme Arduino afin de lire la valeur du capteur de courant.*

Partie 2 : gestion de la mesure de tension

La carte Arduino ne peut pas lire la tension de $24V_{CC}$ fournie par le panneau solaire. Nous allons utiliser le montage "pont diviseur" pour diminuer cette tension :



5. *Retrouvez (démontrez) la formule liant la tension $V2$ à la tension $V1$ (utilisez la loi des mailles pour trouver le courant I puis la loi d'Ohm pour trouver $V2$. La formule ne doit dépendre que de $V1$, $R1$ et $R2$).*
6. *On a choisi $R1=6,6k\Omega$. En déduire la valeur de $R2$ pour obtenir $5V$ (sur $V2$) quand $V1$ vaut $24V$.*
7. *A partir de votre formule, complétez le tableau suivant :*

V1 (volts)	24	18	10	5	0
V2 (volts)					

8. *Sachant que c'est l'entrée analogique A0 qui est utilisée, faites le programme Arduino afin de lire la valeur de cette tension.*

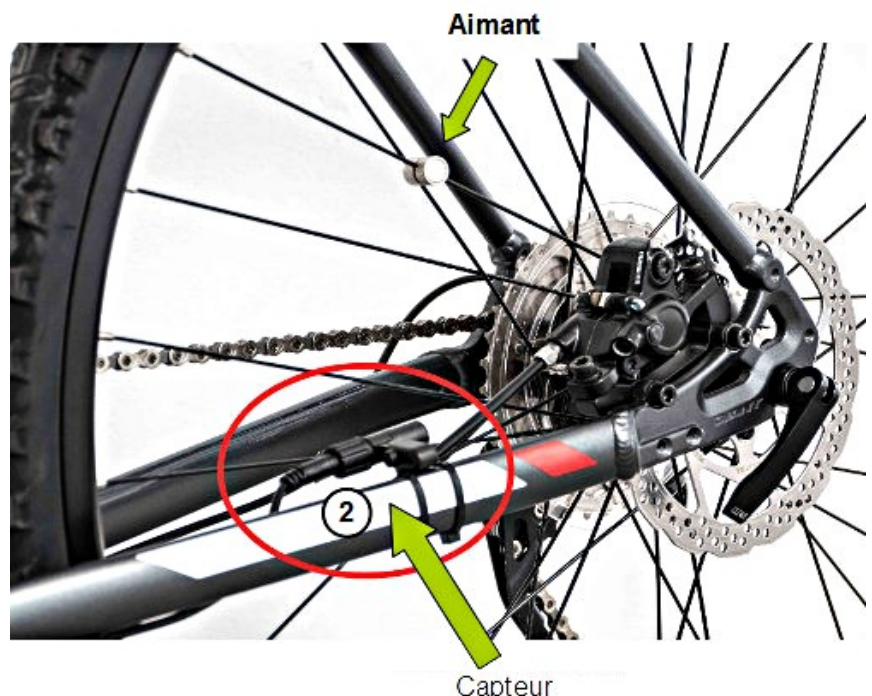
Exercice 10: réalisation du programme de calcul de la vitesse d'un cycliste

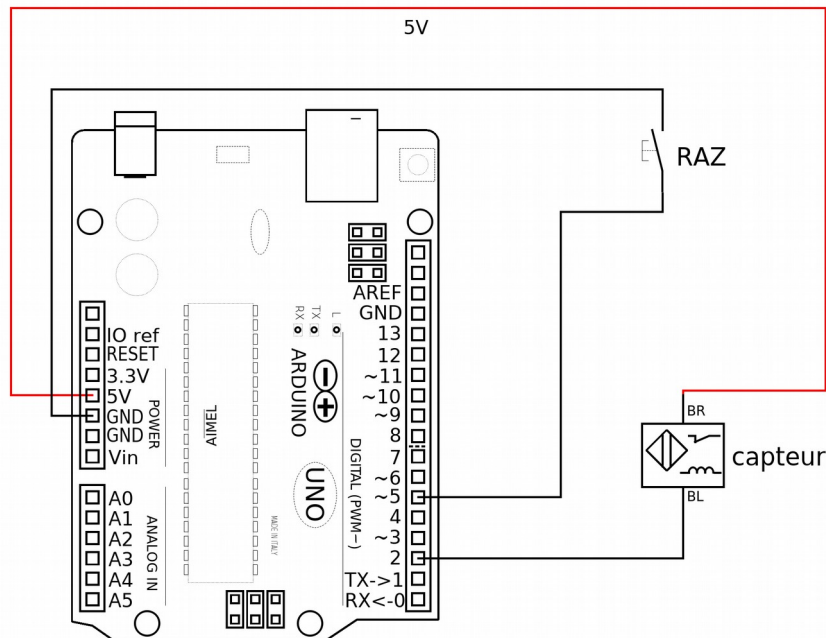
Le vélo dispose d'un capteur (aimant+capteur), placé sur la roue (26 pouce = 665 mm de diamètre avec le pneu).

A chaque passage de l'aimant devant le capteur, celui-ci est activé.

Nom des variables à utiliser dans le programme :

- **etat_BP** (lecture du BP RAZ)
- **distance_km** (distance parcourue en km)
- **vitesse_kmh** (vitesse en km/h)





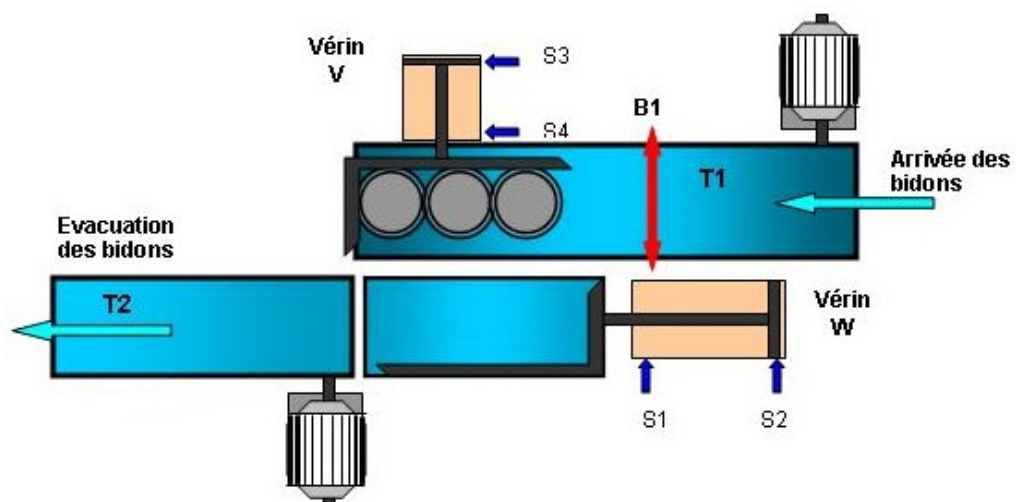
1. En vous aidant du câblage, faites le programme Arduino afin de lire l'état du capteur et de calculer la distance parcourue. La distance est remise à zéro par l'appui sur le bouton poussoir RAZ. Attention : quand le capteur est activé (à "1" logique) on ne compte qu'une seule impulsion (à base vitesse une impulsion dure plusieurs cycles du programme Arduino).
2. Rajoutez le calcul de la vitesse (faire le calcul toutes les 15 secondes en utilisant la fonction `Millis()`).
3. Calculez le nombre d'impulsion par seconde que reçoit la carte Arduino si le vélo roule à 40 km/h. Quel est le risque s'il les impulsions sont trop rapides ?

Exercice 11: réalisation d'un automate séquentiel à l'aide du langage C (Arduino)

Équipement d'emballage de bidons

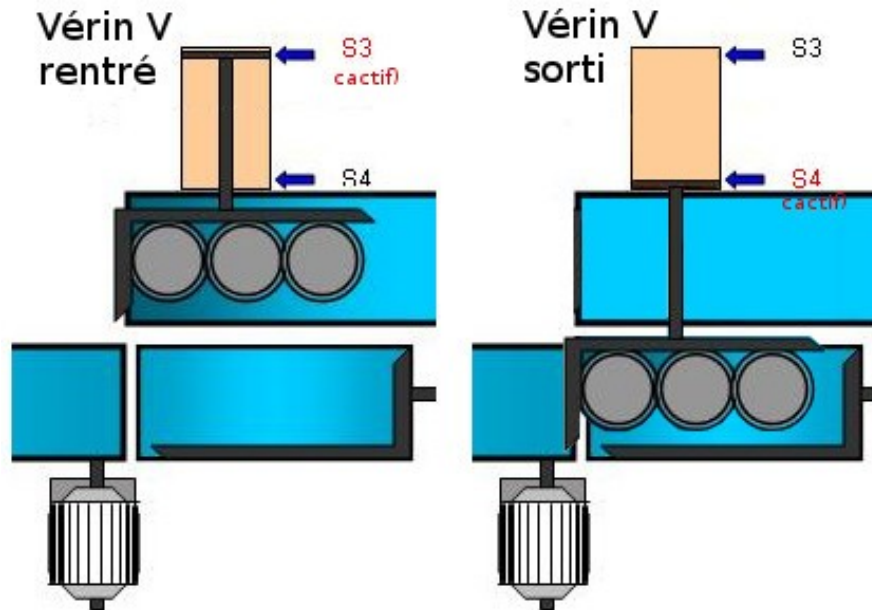
schéma vu de dessus →

L'équipement est utilisé pour former des lots de 3 bidons.
La détection des bidons est assurée par un capteur photo-électrique "B1".



Fonctionnement d'un vérin : un vérin est un élément pneumatique (ou hydraulique) qui permet un déplacement (il peut pousser, tirer, ...). Chaque vérin dispose ici de 2 capteurs TOR (Tout Ou Rien) indiquant la position de celui-ci (rentré ou sorti).

Exemple des 2 positions possibles du vérin V (capteurs S3 et S4):



Tableaux des capteurs et actionneurs

capteurs	désignation	Câblage Arduino (N° de pin)	actionneurs	désignation	Câblage Arduino (N° de pin)
DCY	BP de départ du cycle	0	T1	Marche tapis T1	10
AT	BP d'arrêt du cycle	1	T2	Marche tapis T2	11
B1	Actif lorsque 3 bidons sont présents	2	V+	Sortie du vérin V	12
S1	Vérin W sorti	3	V-	Rentrée du vérin V	13
S2	Vérin W rentré	4	W+	Sortie du vérin W	14
S3	Vérin V rentré	5	W-	Rentrée du vérin W	15
S4	Vérin V sorti	6			

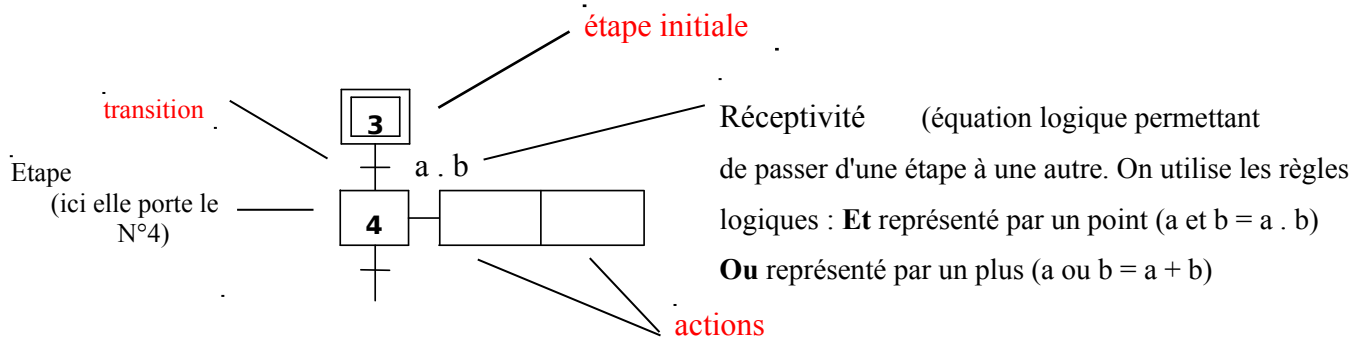
Description du fonctionnement :

Une impulsion sur le bouton poussoir DCY permet le démarrage de l'équipement. On met en marche le tapis T1 et les bidons arrivent par le tapis T1 et sont acheminés devant le vérin V. Lorsque le nombre est atteint (capteur B1 actif), le tapis s'arrête et le transfert des bidons s'effectue par les différents vérins (on sort le vérin V puis on le rentre puis on sort le vérin W puis on le rentre. A chaque action on vérifie l'état du capteur associé à l'action du vérin. Lorsque les bidons sont arrivés sur le tapis T2 on met en route celui 10 secondes. Ensuite le cycle recommence.

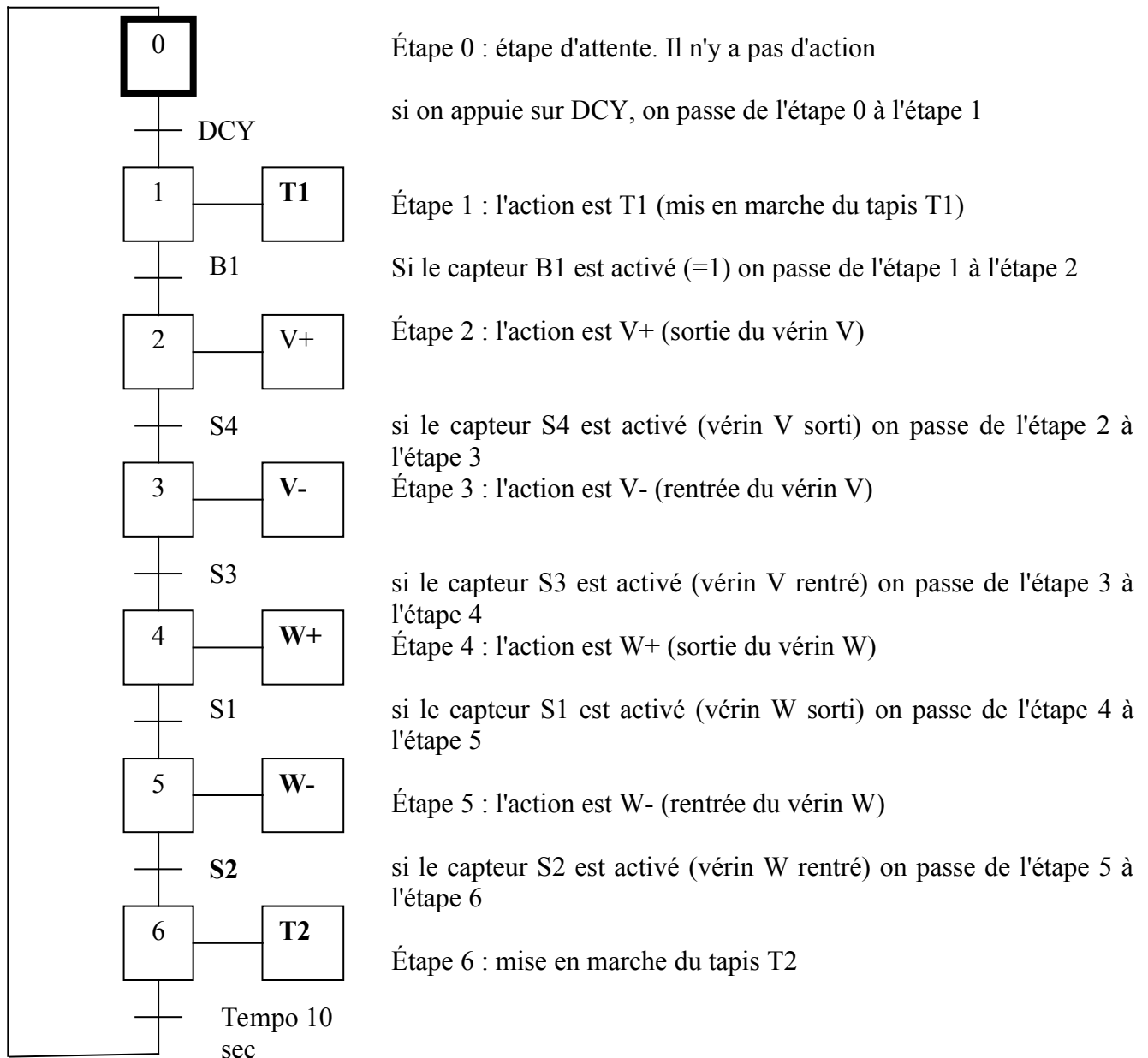
Ce genre de système est dit séquentiel (les actions s'enchainent les unes après les autres). Pour les automaticiens il existe un système de programmation prévu spécialement pour ça. Cela s'appelle le GRAFCET.

Il est composé:

- d'**étapes** auxquelles sont associées des **actions**.
- de **transitions** auxquelles sont associées des **réceptivités**.
- de liaisons orientées entre les étapes et les transitions (du haut vers le bas en général).



Voilà le résultat du grafcet (avec explications à droite de celui-ci):



La difficulté va être d'adapter quelque chose de séquentiel à un programme informatique !

La solution proposée (il en existe d'autres bien sûr) est de créer une variable « étape » qui aura comme valeur le numéro de l'étape de notre automatisme. Ainsi si « étape » vaut 3, on est dans l'étape 3 (avec comme action V-).

Voici la proposition de programme Arduino (Mega)

```
/***/ debut du programme */***/

// définition des pins
#define dcy_pin 0
#define at_pin 1
#define b1_pin 2
#define s1_pin 3
#define s2_pin 4
#define s3_pin 5
#define s4_pin 6
#define t1_pin 10
#define t2_pin 11
#define vplus_pin 12
#define vmoins_pin 13
#define wplus_pin 14
#define wmoins_pin 15

// declaration des variables
boolean dcy,at, b1, s1, s2, s3, s4;
int etape=0;
unsigned long temps1;

void setup() {

  // affectation des pins
  pinMode(dcy_pin, INPUT); // definition de l'entree dcy //
  digitalWrite(dcy_pin, HIGH); // activation du pullup de la broche en entrée
  pinMode(at_pin, INPUT); // definition de l'entree at //
  digitalWrite(at_pin, HIGH); // activation du pullup de la broche en entrée
  pinMode(b1_pin, INPUT); // definition de l'entree b1 //
  pinMode(s1_pin, INPUT); // definition de l'entree s1 //
  pinMode(s2_pin, INPUT); // definition de l'entree s2 //
  pinMode(s3_pin, INPUT); // definition de l'entree s3 //
  pinMode(s4_pin, INPUT); // definition de l'entree s4 //
  pinMode(t1_pin, OUTPUT); // definition de t1 en sortie //
  pinMode(t2_pin, OUTPUT); // definition de t2 en sortie //
  pinMode(vplus_pin, OUTPUT); // definition de vplus en sortie //
  pinMode(vmoins_pin, OUTPUT); // definition de vmoins en sortie //
  pinMode(wplus_pin, OUTPUT); // definition de wplus en sortie //
  pinMode(wmoins_pin, OUTPUT); // definition de wmoins en sortie //
  etape=0; //initialisation de l'automatisme
}

void loop(){

  // lecture des entrées (boutons poussoirs et capteurs
  dcy = digitalRead(dcy_pin); // lecture du bouton poussoir dcy//
  at = digitalRead(at_pin); // lecture du bouton poussoir at//
  b1 = digitalRead(b1_pin); // lecture du capteur b1//
  s1 = digitalRead(s1_pin); // lecture du capteur s1//
  s2 = digitalRead(s2_pin); // lecture du capteur s2//
  s3 = digitalRead(s3_pin); // lecture du capteur s3//
  s4 = digitalRead(s4_pin); // lecture du capteur s4//

  2_prototypage_arduino_eleves (bonus).odt
```

```
// Gestion de la structure de l'automatisme (passage d'une étape à une autre)
```

```
if (etape==0 && dcy==HIGH)
{
    etape=1;
}
if (etape==1 && b1==HIGH)
{
    etape=2;
}
if (etape==2 && s4==HIGH)
{
    etape=3;
}
if (etape==3 && s3==HIGH) etape=4;

if (etape==4 && s1==HIGH)
{
    etape=5;
}
if (etape==5 && s2==HIGH)
{
    etape=6;
    temps1=millis(); // memorisation du temps à la validation de l'étape 6
}
if (etape==6 && (millis()-temps1)>10000) // attente de 10s
{
    etape=0;
}
```

```
// gestion des sorties (actions)
```

```
if(etape==1)
{
    digitalWrite(t1_pin, HIGH);
}
else
{
    digitalWrite(t1_pin, LOW);
}
if(etape==2)
{
    digitalWrite(vplus_pin, HIGH);
}
else
{
    digitalWrite(vplus_pin, LOW);
}
if(etape==3)
{
    digitalWrite(vmoins_pin, HIGH);
}
else
{
    digitalWrite(vmoins_pin, LOW);
}
if(etape==4) digitalWrite(wplus_pin, HIGH);
else digitalWrite(wplus_pin, LOW);
if(etape==5)
{
    digitalWrite(wmoins_pin, HIGH);
}
else
```

```

{
  digitalWrite(wmoins_pin, LOW);
}
if(etape==6)
{
  digitalWrite(t2_pin, HIGH);
}
else
{
  digitalWrite(t2_pin, LOW);
}

} // fin de la boucle loop

```

1. *Faire l'algorithme (sous forme d'algorithmes) du programme*
2. *Expliquez, en détail, ce que va faire le programme et comment il le fait.*

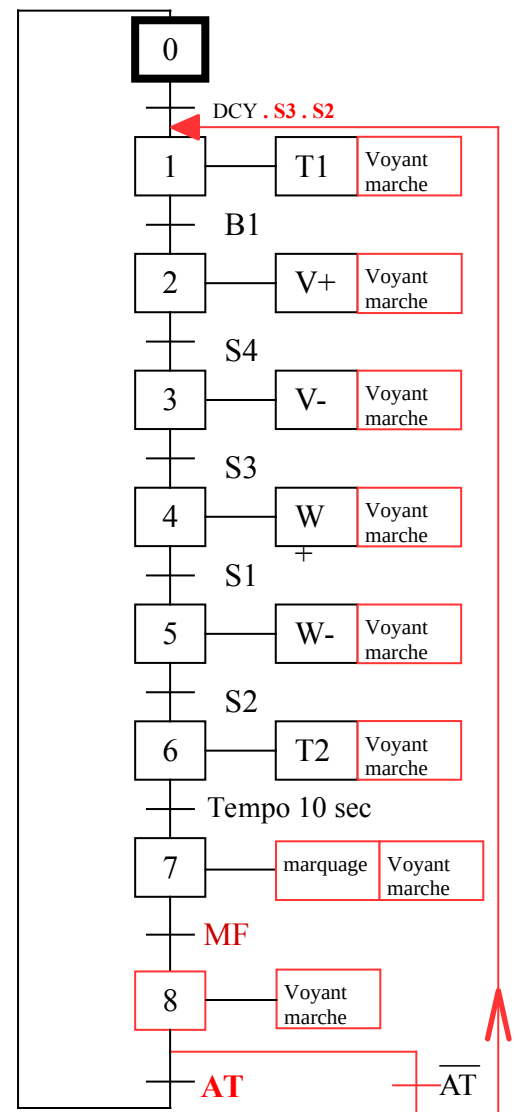
Suite à des essais, la machine et le grafset sont modifiés. On rajoute la fonction marquage réalisée par un tampon qui écrit sur les 3 bidons en même temps. Un capteur indique quand le marquage est réalisé (MF). Une lampe est rajoutée pour indiquer que la machine est en marche.

De plus à la fin du cycle il y a 2 possibilités : soit on appuie sur AT et le cycle s'arrête (on va à l'étape 0), soit on n'appuie pas sur AT et le cycle recommence (on va à l'étape 1).

Capteurs et actionneurs rajoutés :

capteurs	désignation	Câblage Arduino (N° de pin)
MF	Marquage Fait	7
actionneurs	désignation	Câblage Arduino (N° de pin)
VM	Voyant « marche »	16
MARQ	marquage	17

3. *Modifiez le programme Arduino pour qu'il réalise le grafset modifié ci-contre:*



Partie 4 : création d'algorithmes ou d'algorigrammes

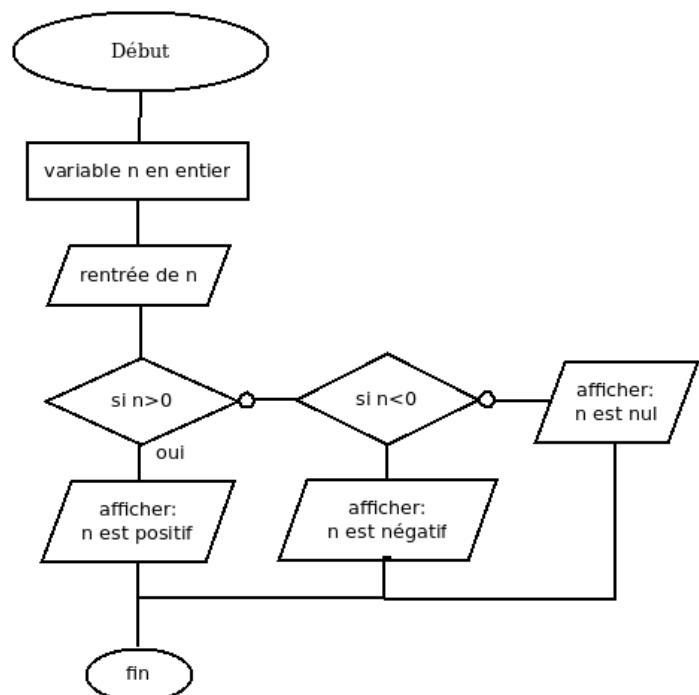
Remarque : pour les exercices suivants vous aurez à créer des programmes informatiques sous la forme d'algorithmes. Jusqu'à présent on a essentiellement travaillé avec la forme graphique (algorigrammes) donc vous pouvez continuer à utiliser cette forme là (mais si vous voulez vous pouvez utiliser la forme normale : SI SINON).

Exemple : un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est positif ou négatif.

Sous la forme « normale » cela donne:
(rq : les variables sont déclarée avant 'début')

```
Variable n en Entier  
Début  
Ecrire "Entrez un nombre : "  
Lire n  
Si n > 0 Alors  
  Ecrire "Ce nombre est positif"  
Sinon  
  Si n < 0 Alors  
    Ecrire "Ce nombre est négatif"  
  Sinon  
    Ecrire "Ce nombre est nul"  
Finsi  
Finsi  
Fin
```

sous la forme d'algorigramme :



Exercice 1 :

Écrivez un algorithme qui échange la valeur de deux variables. Par exemple si a=3 et b=12, a sera égal à 12 et b à 3.

Exercice 2 : Jeu du nombre mystérieux (à deviner)

Ce jeu se joue de la manière suivante :

L'ordinateur a choisi un nombre entier au hasard entre 0 et 100.

Le joueur rentre un nombre. Le programme lui répond si son choix est plus grand ou plus petit que le nombre recherché et redemande au joueur de rentrer un nouveau nombre etc

Si le joueur rentre le bon nombre, il a gagné (le programme affiche le nombre d'essais effectués pour arriver au nombre recherché). Le programme lui demande alors s'il veut rejouer. S'il ne veut pas, le programme s'arrête.

Écrivez un algorithme de ce programme.

Exercice 3 : nombre pair ou impair ?

Écrire un algorithme qui demande un nombre entier à l'utilisateur et indique si ce nombre est pair ou impair.

Astuce : un entier est pair, par exemple, si en lui retranchant suffisamment de fois 2 on arrive à 0
ou

Astuce N°2: on peut utiliser les propriétés mathématiques de la division "entière".

Exercice 4 :

La croissance économique annuelle du PIB d'un pays est 2 %.

Écrire un algorithme qui permet de déterminer le nombre d'années au bout desquelles le PIB (produit intérieur brut) aura doublé.