

NOM : PRÉNOM :	CLASSE :
-------------------------------------	-----------------

CONDITION DE RÉALISATION :

Travail individuel **Durée : 3 heures**

Matériel :

- un ordinateur sous Ubuntu avec le logiciel Arduino
- une maquette Arduino Uno
- une maquette shield interrupteurs et leds

Documents :

- le sujet du TP
- le cours d'algorithmie

L'objectif de ce TP est de **programmer un micro-contrôleur Arduino** en utilisant les différentes structures algorithmiques vues en cours. Le TP est effectué sous Ubuntu avec le logiciel Arduino.

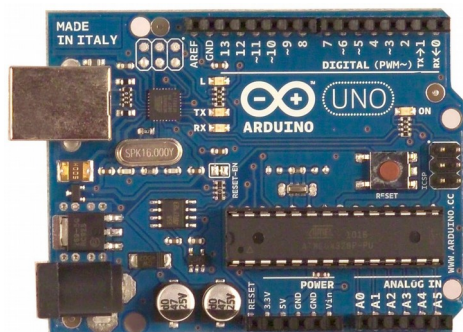
Introduction

Les maquettes Arduino UNO utilisées en TP sont basées sur un micro-contrôleur AVR ATmega328P. Vous allez dans ce TD découvrir ce composant et les possibilités qu'il offre en programmation.

Pour compléter le document, utilisez la documentation du micro-contrôleur disponible sur le réseau et éventuellement de recherche sur internet.

- ✓ Rappelez la différence entre un microprocesseur et un micro-contrôleur

- ✓ Retrouvez les éléments suivants sur la carte Arduino : connecteur USB, quartz 16MHz, connecteur d'alimentation, micro-contrôleur ATmega328P et bouton Reset.



(PCINT14/RESET) PC6	1	28	<input type="checkbox"/>	PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD) PD0	2	27	<input type="checkbox"/>	PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXD) PD1	3	26	<input type="checkbox"/>	PC3 (ADC3/PCINT11)
(PCINT18/INT0) PD2	4	25	<input type="checkbox"/>	PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1) PD3	5	24	<input type="checkbox"/>	PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0) PD4	6	23	<input type="checkbox"/>	PC0 (ADC0/PCINT8)
VCC	7	22	<input type="checkbox"/>	GND
GND	8	21	<input type="checkbox"/>	AREF
(PCINT6/XTAL1/TOSC1) PB6	9	20	<input type="checkbox"/>	AVCC
(PCINT7/XTAL2/TOSC2) PB7	10	19	<input type="checkbox"/>	PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1) PD5	11	18	<input type="checkbox"/>	PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0) PD6	12	17	<input type="checkbox"/>	PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1) PD7	13	16	<input type="checkbox"/>	PB2 (SS/OC1B/PCINT2)
(PCINT0/CLKO/ICP1) PB0	14	15	<input type="checkbox"/>	PB1 (OC1A/PCINT1)

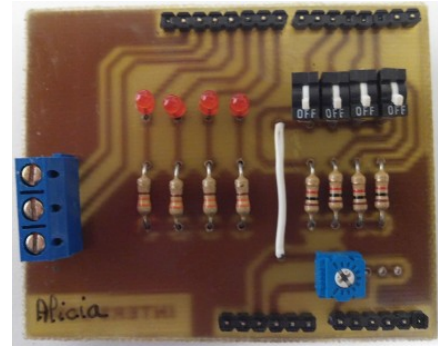
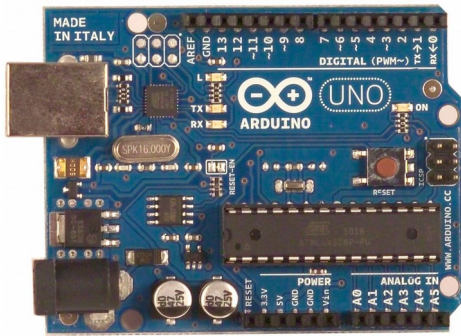
Le micro-contrôleur ATmega328P, fabriqué par est un micro-contrôleur bits : cela signifie que son bus de a une largeur de bits.

Le modèle utilisé est en boîtier de broches. L'alimentation VCC se retrouve sur la broche et la masse sur les broches

La plage d'alimentation du composant (VCC) va de V à V.

1. Présentation des cartes

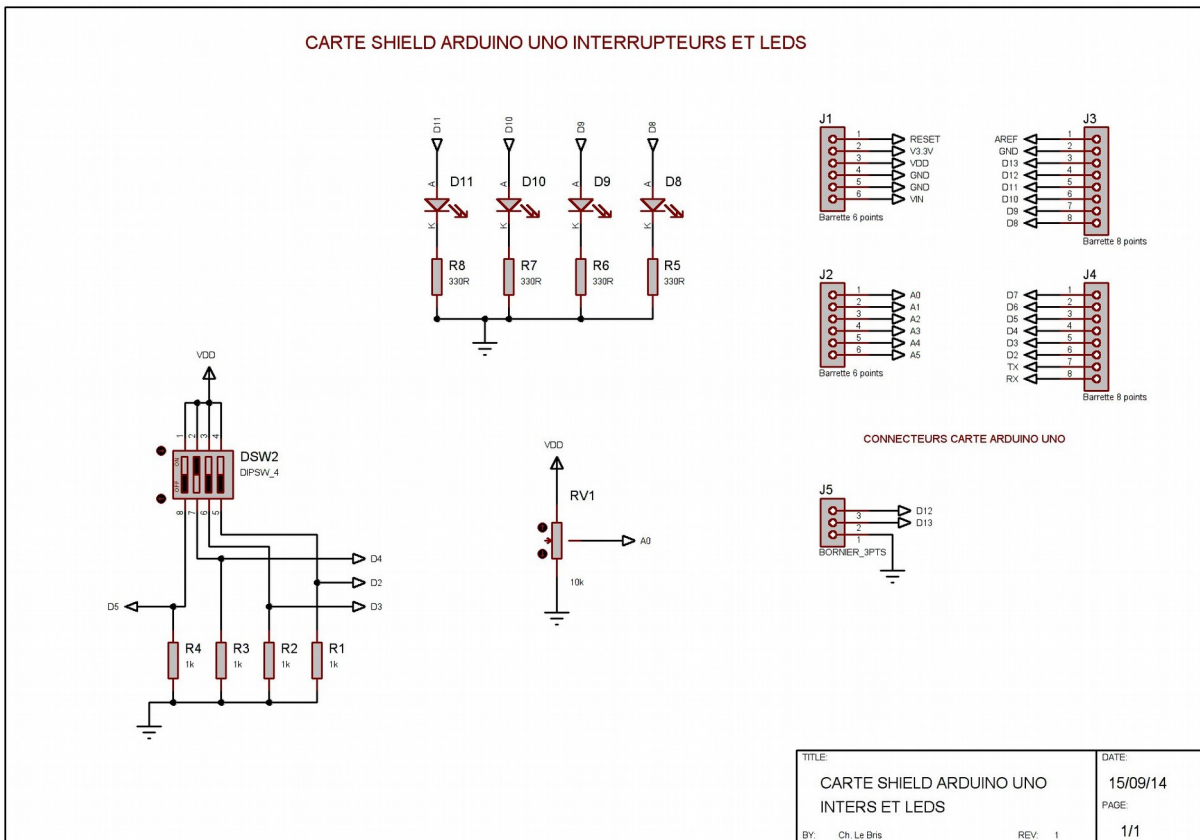
On utilise la maquette Arduino Uno et la maquette shield, encore appelée carte fille, interrupteurs et leds.



Cette carte shield est composée de 4 leds, 4 interrupteurs, un potentiomètre et un bornier connecté à la liaison série de l'Arduino.

- ✓ Hors tension connectez la carte shield « interrupteurs leds » sur la carte Arduino. Prenez soin de ne pas casser des broches !
- ✓ Démarrez votre PC sous Ubuntu. Le compte à utiliser est **sti2dsin**, le mot de passe est **sti2dsin**.
- ✓ Connecter l'Arduino au PC avec un câble USB. Vous pouvez démarrer !

Le schéma de la carte Shield est le suivant :



- ✓ A partir de ce schéma, compléter le tableau ci-dessous en notant les interrupteurs I2 à I5, les leds D8 à D11 et le potentiomètre RV1.

Numéro de broche	2	3	4	5	8	9	10	11	A0
composant relié	inter I2				led D8				

2. Allumer les leds :

On donne ci-dessous le programme d'allumage de la led D1 connectée sur la broche 13 de l'Arduino :



Commentaires : facilitent la compréhension du programme lors d'une réouverture ultérieure, en cas de travail en équipe, ...
Les commentaires s'écrivent entre « /* » et « */ » ou bien derrière « // » pour un commentaire sur une seule ligne.

Configuration des entrées/sorties : les broches numériques de l'Arduino peuvent être utilisées en entrée ou en sortie. Ici on configure la broche 8 en sortie (Output).

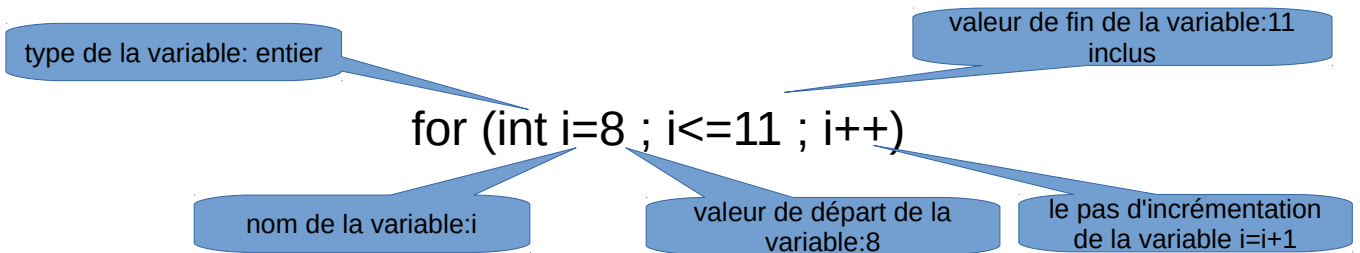
Programme principal : boucle infinie dans laquelle les opérations s'effectuent dans l'ordre d'écriture. Ici on écrit un niveau logique haut sur la sortie 8 de l'Arduino.
Chaque instruction se termine par un point-virgule.

```
/*
 nom: led.ino
 but: programme d'allumage d'une led
 auteur: lycee f ledantec Lannion
 date: 10 janvier 2018
 */
void setup() {
 // initialise les entrées/sorties
 pinMode(8,OUTPUT);
}
void loop() {
 // allume la led D8
 digitalWrite(8,HIGH);
}
```

Zone des messages d'erreurs ou succès envoyés par le programme.

- ✓ Démarrez l'environnement de développement Arduino et écrivez le programme donné page précédente.
- ✓ Compilez votre programme , vérifiez l'absence d'erreur et téléchargez le programme  dans l'Arduino. Votre programme s'exécute dès la fin du téléchargement. Vérifiez le bon fonctionnement, la led D8 doit s'allumer. et **faire valider par le professeur**. Validation prof :
- ✓ Modifiez votre programme pour allumer les leds D8, D10, en laissant éteinte D9, D11. Vérifiez le bon fonctionnement et faites valider par le professeur. Validation prof :
- ✓ L'instruction `delay(x)` permet de faire une temporisation (une attente) de x millisecondes. Écrivez un programme qui fait clignoter la led D8 au rythme de la seconde. Validation prof :
- ✓ Écrivez un programme qui crée un chenillard (défilement de gauche à droite puis de droite à gauche) sur les leds D8 à D11 avec des temporisations de 200ms entre deux changements d'états. Validation prof :

Votre programme du chenillard est probablement long et très répétitif. Nous allons utiliser une boucle de type **pour** (**for** en anglais) afin d'allumer ou d'éteindre les leds.



On vous donne ci-dessous le programme d'un chenillard effectuant un défilement aller (droite à gauche), utilisant cette boucle **for**.

- ✓ Écrivez ci-dessous l'algorithme correspondant à ce programme.

```

/*
 nom:allerChenillard.ino
 but: programmer un allert d'un chenillard
 auteur: lycee F Ledantec Lannion
 date:10 janvier 2018
 */
void setup(){
 //pour i un entier allant de 8 à 11 par pas de 1
 for (int i=8 ; i<=11 ; i++){
 //initialise les sorties i
 pinMode(i,OUTPUT);
 }
 }
void loop(){
 //pour i un entier allant de 8 à 11 par pas de 1
 for (int i=8 ; i<=11 ; i++){
 //allume la led i
 digitalWrite(i,HIGH);
 delay(200);
 digitalWrite(i,LOW);
 }
 }

```

Validation prof :

- ✓ Écrivez ce programme sous Arduino, complétez-le afin d'y ajouter la boucle retour du chenillard. Attention les leds des extrémités ne doivent pas rester allumées plus longtemps que les autres.

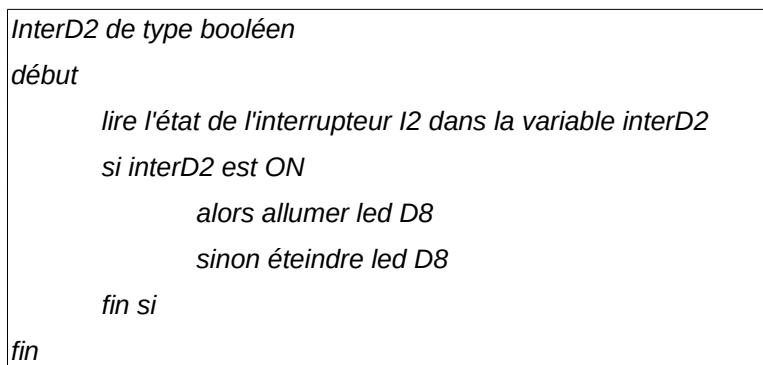
Validation prof :

3. Lire une entrée numérique

Afin de lire l'état d'un interrupteur il est obligatoire de **déclarer une variable** dans laquelle sera stocké le résultat de cette lecture. Dans notre cas si l'interrupteur est OFF le contenu de la variable sera 0 (niveau logique bas) et inversement 1 (niveau logique haut) si l'interrupteur est ON.

Une variable de type booléen, **boolean** en anglais, est suffisante pour la lecture d'une entrée. Pour déclarer cette variable on écrira l'instruction : **boolean interD2 ; // déclarer une variable I1 de type booléen**

On donne ci-dessous l'algorithme du programme principal permettant d'allumer la led D8 si l'interrupteur Inter 2 est ON.

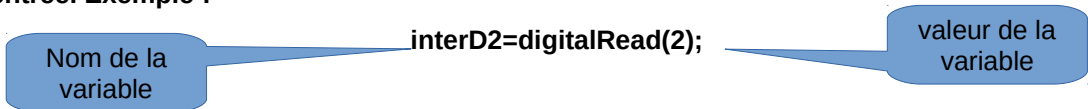


```

/*
 nom: lectureinterD2.ino
 but: programme lecture interD2
 auteur: lycee f ledantec Lannion
 date: 10 janvier 2018
 */
boolean interD2;
void setup() {
 // initialise les entrées/sorties
 pinMode(8,OUTPUT);
 pinMode(2,INPUT);
 }
void loop() {
 //lire l'etat de la broche 2 reliée à interD2 et stocker
 //la valeur lue dans la variable interD2*
 interD2=digitalRead(2);
 if (inter 2==HIGH) {
 if (interD2==HIGH){
 //allume la led D8
 digitalWrite(8,HIGH);
 }
 else digitalWrite(8,LOW); //éteinte la led D8
 }
 }

```

L'instruction qui permet la lecture de l'état d'une entrée numérique est **digitalRead(X)** ou X est le numéro de l'entrée. Exemple :



- ✓ Écrivez le programme correspondant à l'algorithme ci-dessus.

Validation prof :

- ✓ Écrivez un programme allumant toutes les leds si l'interrupteur I2 est ON. Vous utiliserez des boucles pour allumer ou éteindre toutes les leds.

Validation prof :

- ✓ Écrivez un programme qui recopie les états des inters I2 à I5 sur les leds D8 à D11. Vous utiliserez une boucle.

Validation prof :

- ✓ Écrivez un programme allumant la led D8 seulement si un nombre pair d'interrupteurs est ON. L'opérateur modulo %, qui calcule le reste de la division entière, vous permettra de détecter un nombre pair.

Validation prof :

- ✓ Modifiez le chenillard aller-retour du 3. afin de commander la vitesse de défilement des leds en fonction des états des interrupteurs I3 à I5. Plus le nombre d'interrupteurs ON est grand, plus lentement défile le chenillard. L'interrupteur I2 permet d'arrêter le chenillard (utilisez une boucle **Tant que : while()**).

Validation prof :

4. Lire une entrée analogique

Le potentiomètre RV1 sur l'entrée analogique A0 de l'Arduino permet de faire varier la tension sur cette entrée de 0 à 5V. Un **convertisseur Analogique-Numérique**, interne à l'Arduino, **convertit cette tension en une valeur numérique de 0 à 1023**.



Nous allons utiliser le moniteur série du PC pour afficher le résultat de la conversion de la tension présente sur la broche A0. On vous donne le programme ci-dessous :

```

/*
  nom: lecturePotar.ino
  but: programme lecture potar
  auteur: lycee f ledantec Lannion
  date: 10 janvier 2018
*/
boolean interD2; int potar;
void setup() {
  // initialise les entrées/sorties
  pinMode(0, INPUT);
  //ouverture du port série vitesse 9600bauds
  Serial.begin(9600);
}

void loop() {
  /*lire l'etat de la broche 2 reliée à interD2 et stocker
  la valeur lue dans la variable interD2*/
  potar=analogRead(0);
  //sans sauter de ligne écrire un message sur le moniteur serie
  Serial.print("valeur lue sur Potar: ");
  /*écrire sur le port série la valeur lue sur potar
  en sautant des lignes*/
  Serial.println(potar);
}

```

- ✓ Écrivez ce programme qui affiche sur le moniteur série la valeur numérique correspondant à la tension analogique sur la broche A0. Testez en faisant varier la position du potentiomètre. Le moniteur série s'ouvre en cliquant sur le bouton à droite *Moniteur série*.

La plage de variation du résultat de la conversion analogique-numérique fournie par l'Arduino peut ne pas convenir. Il est possible d'effectuer un **réétalonnage** en utilisant la fonction de mise à l'échelle :

`map(x, borne inf de x, borne sup de x, nouvelle borne mini de x, nouvelle borne maxi de x) ;`

Par exemple pour obtenir une valeur de potar qui varie entre 0 et 1024 rétalonnée entre 0 et 100 on écrira :

`float X= map(potar,0, 1024, 0, 100) ;`

- ✓ Modifiez le programme précédent afin d'obtenir une valeur sur le port série variant entre 0 et 100.

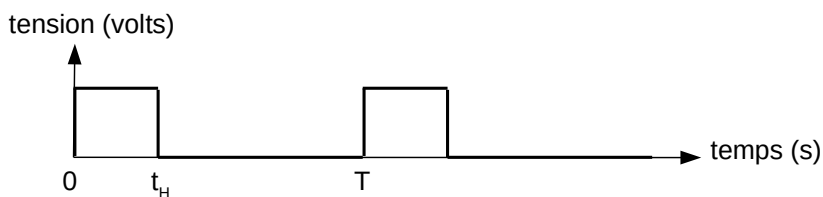
Validation prof :

- ✓ Modifiez le chenillard du 3. afin de commander la vitesse de défilement des leds en fonction de la position du potentiomètre.

Validation prof :

5. Écrire sur une sortie analogique

Une sortie analogique correspond à un signal PWM (Pulse Width Modulation) ou MLI en français (Modulation de largeur d'impulsion). L'allure de ce type de signal est la suivante :



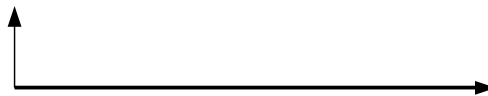
t_H : temps à l'état haut du signal

T : période du signal

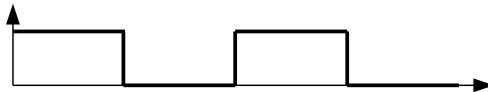
$\alpha = t_H / T$: rapport cyclique du signal

Il est possible d'obtenir différentes allures du signal en modifiant la valeur de t_H , durée du temps à l'état haut. Le schéma ci-dessous montre 3 allures du signal pour 3 valeurs différentes de t_H .

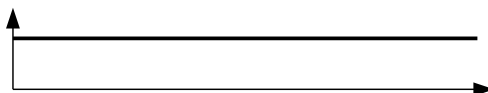
`AnalogWrite(9,0) ;`



`AnalogWrite(9,127) ;`



`AnalogWrite(9,255) ;`



- ✓ Écrivez un programme qui allume la led D9 avec une intensité fonction de la position du potentiomètre. Vous utiliserez la fonction PWM sur la broche 9.

Validation prof :