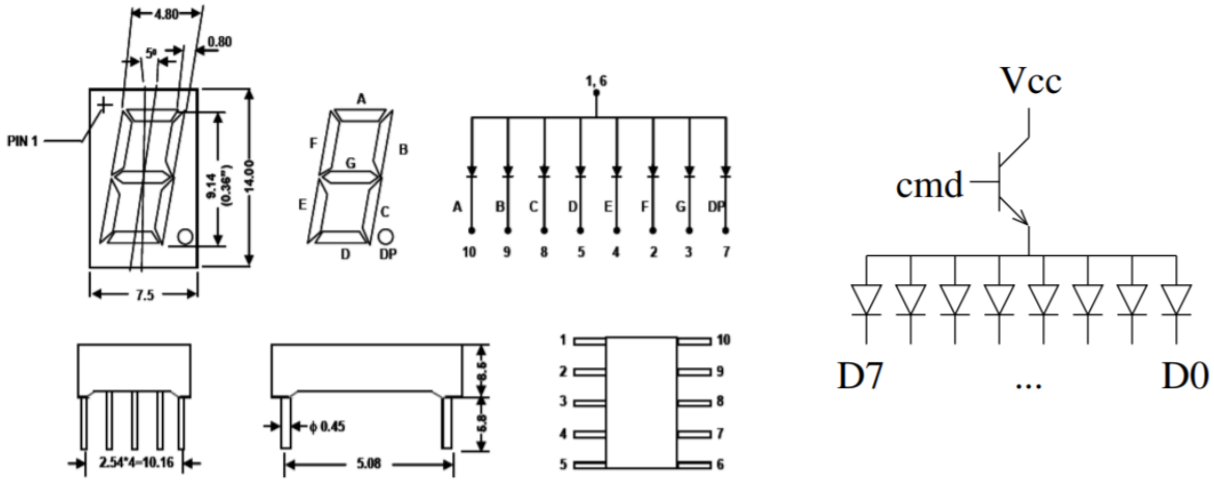
Afficheur 7 Segments

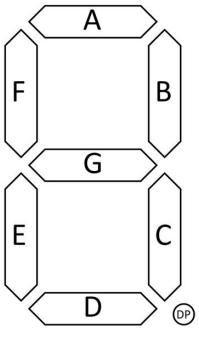
Consignes :
Pour ce TP, vous ferez le compte rendu des questions sur un éditeur de texte sur votre PC.
Un compte-rendu par personne.

Un afficheur 7 segments

Un afficheur 7-segments (Fig. ci-dessous) est formé d'un ensemble de 8 diodes (7-segments et un point décimal) que nous manipulons en plaçant le GPIO du microcontrôleur à l'état haut ou bas.

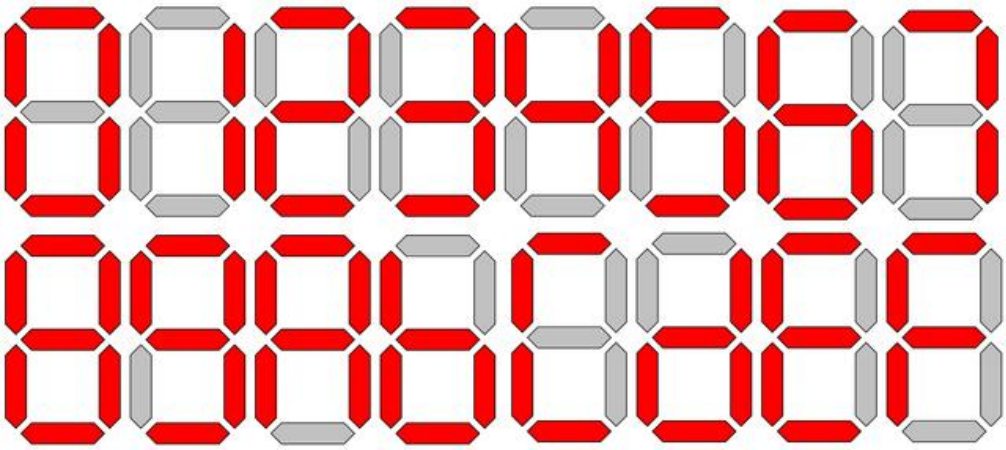


Les afficheurs 7 segments sont constitués de 7 segments, d'où leur nom. Ces segments sont nommés A, B, C, D, E, F et G par convention, et ils se présentent dans l'ordre illustré ci-dessous.



Chaque segment correspond à une LED qu'il est possible d'allumer ou d'éteindre pour former des chiffres, des lettres et même des caractères spéciaux rudimentaires. En général, les afficheurs disposent de 7 segments et d'un "point décimal" qui peut être utilisé pour afficher des nombres à virgule ou des sous-unités (dixième de seconde par exemple).

Il existe une multitude de couleurs d'afficheurs 7 segments : rouge, vert, jaune, orange, bleu, blanc, etc. Il existe une multitude de tailles, du petit afficheur de quelques millimètres de côté à plusieurs dizaines de centimètres.



Chiffres de 0 à 9 et lettres de A à F

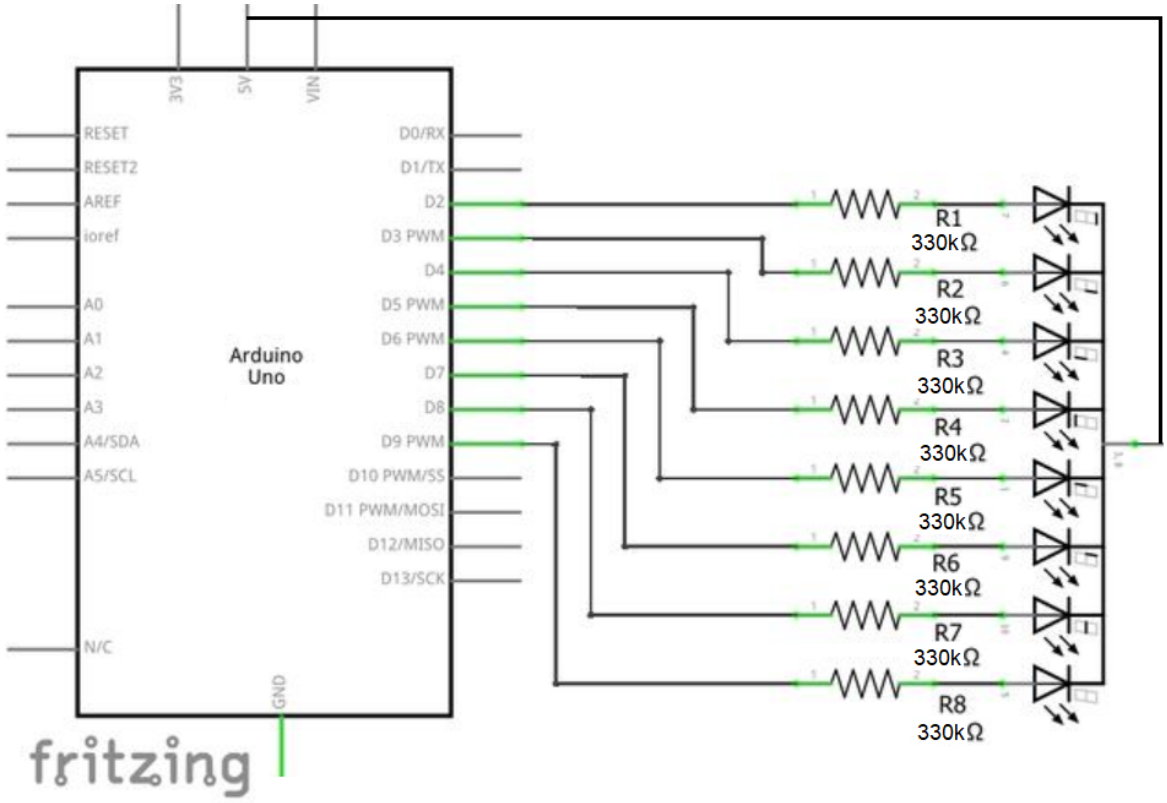
Afficher des chiffres avec un afficheur 7 segments se résume à allumer les LEDs des segments adéquats.

1. Remplissez le tableau d'agencement des segments et l'état des segments pour chaque chiffre (ON/OFF)

Digit	G	F	E	D	C	B	A
0							
1							
2							
3							
4							
5							
6							
7							
8							
9							
A							
B							
C							
D							
E							
F							

Le montage

2. En utilisant TINKERCAD (https://www.tinkercad.com), vous allez reproduire le branchement d'une carte Arduino et d'un afficheur 7 segments à l'aide du schéma électrique suivant :



Pour cela, vous utiliserez sur TINKERCAD (section « circuit ») :

- Un arduino UNO
- Un afficheur 7 segments (ANODE commune)
- Une breadboard (plaque d'essais)
- Des résistances (330kΩ)
- Des fils

Branchements des segments sur les broches de l'Arduino

Segment	Broche ARDUINO (pin)
A	2
B	3
C	4
D	5
E	6
F	7
G	8
DP	9

Le code

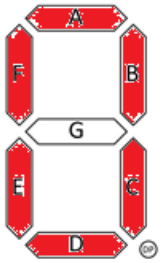
Le but du code d'exemple ci-dessous est d'afficher des chiffres de 0 à F (15 en hexadécimal) en boucle.

```
/* Broches des différents segments de l'afficheur */

const byte PIN_SEGMENT_A = 2;

const byte PIN_SEGMENT_B = 3;

const byte PIN_SEGMENT_C = 4;

const byte PIN_SEGMENT_D = 5;

const byte PIN_SEGMENT_E = 6;

const byte PIN_SEGMENT_F = 7;

const byte PIN_SEGMENT_G = 8;

const byte PIN_SEGMENT_DP = 9;
```

On va commencer ce code de manière très classique avec les diverses déclarations de constantes pour les broches.

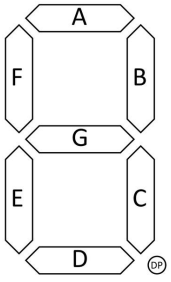
3. Donnez la correspondance valeur -> état des segments dans le code (Aidez-vous du tableau d'agencement).

Exemple :

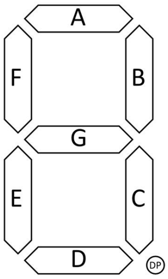


0	b	DP	G	F	E	D	C	B	A
0	b	0	0	1	1	1	1	1	1

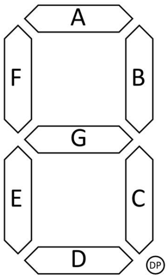
A vous de représenter les caractères de 1 à F (schéma + tableau)



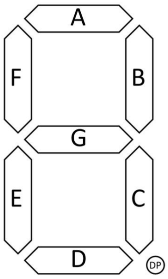
0	b	DP	G	F	E	D	C	B	A
0	b	0							



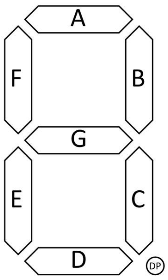
0	b	DP	G	F	E	D	C	B	A
0	b	0							



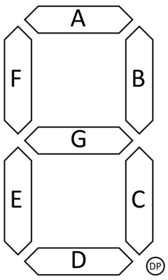
0	b	DP	G	F	E	D	C	B	A
0	b	0							



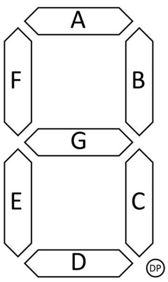
0	b	DP	G	F	E	D	C	B	A
0	b	0							



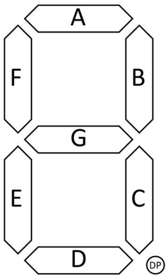
0	b	DP	G	F	E	D	C	B	A
0	b	0							



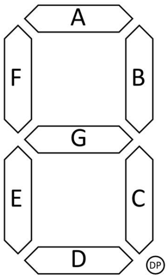
0	b	DP	G	F	E	D	C	B	A
0	b	0							



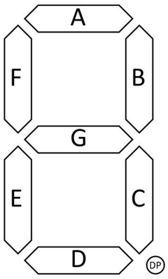
0	b	DP	G	F	E	D	C	B	A
0	b	0							



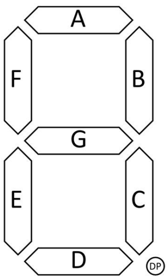
0	b	DP	G	F	E	D	C	B	A
0	b	0							



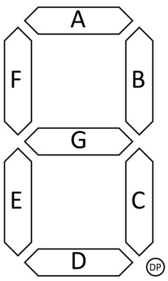
0	b	DP	G	F	E	D	C	B	A
0	b	0							



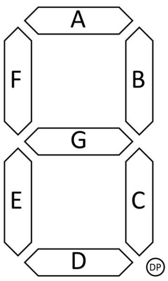
0	b	DP	G	F	E	D	C	B	A
0	b	0							



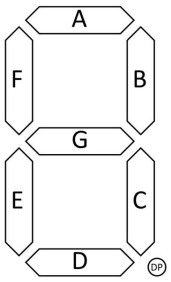
0	b	DP	G	F	E	D	C	B	A
0	b	0							



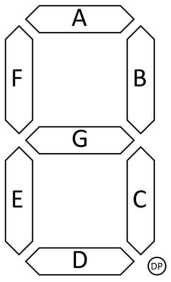
0	b	DP	G	F	E	D	C	B	A
0	b	0							



0	b	DP	G	F	E	D	C	B	A
0	b	0							



0	b	DP	G	F	E	D	C	B	A
0	b	0							



0	b	DP	G	F	E	D	C	B	A
0	b	0							

```
/* Table de correspondance valeur -> états des segments de l'afficheur */
const byte LUT_ETATS_SEGMENTS[] = {
  0b00111111, //……
  0b00000110, //……
  0b01011011, //……
  0b01001111, //……
  0b01100110, //……
  0b01101101, //……
  0b01111101, //……
  0b00000111, //……
  0b01111111, //……
  0b01101111, //……
  0b01110111, //……
  0b01111100, //……
  0b00111001, //……
  0b01011110, //……
  0b01111001, //……
  0b01110001 //……
};
```

On continue ensuite avec une autre constante, un tableau d'octets, qui contiendra notre table de correspondance de chiffre vers segments.

```
/** Fonction setup() */
void setup() {


  /* Toutes les broches en sorties */
  pinMode(PIN_SEGMENT_A, OUTPUT);
  digitalWrite(PIN_SEGMENT_A, LOW);
  pinMode(PIN_SEGMENT_B, OUTPUT);
  digitalWrite(PIN_SEGMENT_B, LOW);
  pinMode(PIN_SEGMENT_C, OUTPUT);
  digitalWrite(PIN_SEGMENT_C, LOW);
  pinMode(PIN_SEGMENT_D, OUTPUT);
  digitalWrite(PIN_SEGMENT_D, LOW);
  pinMode(PIN_SEGMENT_E, OUTPUT);
  digitalWrite(PIN_SEGMENT_E, LOW);
  pinMode(PIN_SEGMENT_F, OUTPUT);
  digitalWrite(PIN_SEGMENT_F, LOW);
  pinMode(PIN_SEGMENT_G, OUTPUT);
  digitalWrite(PIN_SEGMENT_G, LOW);
  pinMode(PIN_SEGMENT_DP, OUTPUT);
  digitalWrite(PIN_SEGMENT_DP, LOW);
}
```

La fonction setup() ne fera pas grand-chose. Quelques pinMode() et quelques digitalWrite() pour initialiser les broches en sortie et à LOW.

```
/** Fonction loop() */
void loop() {
  static byte chiffre = 0;
  static byte etat_dp = 0;

  /* Affiche le chiffre */
  affiche_chiffre_7seg(chiffre, etat_dp);

  /* Incrémente le chiffre de 0 à 15 */
  if (++chiffre == 16) {
    chiffre = 0;
  }

  /* Fait clignoter le point décimal (inverse l'état à chaque fois) */
  etat_dp = !etat_dp;

  /* Délai pour la démo */
  delay(1000);
}
```

La fonction loop() fait quatre choses :

- Elle affiche le chiffre courant au moyen de la fonction affiche_chiffre_7seg(),
- Elle incrémente de 1 la valeur du chiffre, en repartant de 0 si la valeur dépasse 15,
- Elle fait clignoter le point décimal en inversant son état,
- Elle attend une seconde avant de recommencer.

```
/** Fonction permettant d'afficher un chiffre sur un afficheur 7 segments */
void affiche_chiffre_7seg(byte chiffre, byte dp) {


  /* Simple sécurité */
  if (chiffre > 15)
    return; // Accepte uniquement des valeurs de 0 à 15.


  /* Conversion chiffre -> états des segments */
  byte segments = LUT_ETATS_SEGMENTS[chiffre];


  /* Affichage */
  digitalWrite(PIN_SEGMENT_A, !bitRead(segments, 0));
  digitalWrite(PIN_SEGMENT_B, !bitRead(segments, 1));
  digitalWrite(PIN_SEGMENT_C, !bitRead(segments, 2));
  digitalWrite(PIN_SEGMENT_D, !bitRead(segments, 3));
  digitalWrite(PIN_SEGMENT_E, !bitRead(segments, 4));
  digitalWrite(PIN_SEGMENT_F, !bitRead(segments, 5));
  digitalWrite(PIN_SEGMENT_G, !bitRead(segments, 6));
  digitalWrite(PIN_SEGMENT_DP, !dp);
}
```

La fonction affiche_chiffre_7seg() s'occupe de l'affichage.

Elle commence par vérifier que le chiffre est bien compris entre 0 et 15 (inclus). Elle utilise ensuite le tableau de correspondance pour "convertir" le chiffre en une série d'états pour les segments. Pour finir, elle utilise une série de digitalWrite() pour écrire l'état de chaque segment.

L'extraction de l'état de chaque segment est réalisée par la fonction bitRead(). Cette fonction permet d'extraire la valeur d'un bit d'un nombre entier. Ici chaque bit de la valeur en sortie du tableau correspond à un segment.

4. Écrivez le programme précédent, testez-le et décrivez ce qui se passe.

5. Modifiez le programme pour inverser l'ordre d'affichage. Faire valider par le professeur et le copier/coller sur votre compte-rendu.