

Samedis bénévoles spécial Arduino

Workshop n° 2

FICHE F6 - CAPTEURS ET ACTIONNEURS

SOMMAIRE

| | |
|--|----|
| 1. Généralités sur les capteurs | 2 |
| 2. Une démonstration | 2 |
| 3. Un cas pratique d'utilisation d'un capteur : la mesure de distance..... | 4 |
| 3.1. Un peu de technologie | 4 |
| 3.2. Application à la robotique | 5 |
| 3.3. Le code de l'Arduino | 8 |
| 4. Un exemple utile d'actionneur : un afficheur 4 digits à LED | 9 |
| 4.1. Principe d'utilisation | 9 |
| 4.2. Pas à pas de montage d'un afficheur | 10 |
| 5. Application : Coupler un capteur et un actionneur | 20 |
| 6. L'asservissement d'un servomoteur..... | 24 |

1. Généralités sur les capteurs

Les capteurs sont très largement utilisés de façon courante et sont essentiels pour la robotique. Ils sont les yeux, les oreilles, les doigts et dans certains cas peu courants l'odorat et le goût d'un système robotique. L'Arduino sait gérer un grand nombre de capteurs et de nombreuses bibliothèques sont venues simplifier leur utilisation au cours des dernières années.

En effet, là où il fallait faire des calculs de conversion et de mise à l'échelle des signaux reçus, il suffit d'appeler une fonction qui les réalisera à la place et de récupérer un résultat. Comme dans d'autres domaines de l'informatique, la réutilisation de code éprouvé et optimisé joue un rôle de levier très important dans le développement d'applications dont les fonctionnalités peuvent s'étoffer et devenir de plus en plus riches dans un environnement d'exécution de plus en plus compact (pensez à la taille d'un Arduino Nano ou micro).

Il reste trois préoccupations essentielles dans l'utilisation des capteurs : leur robustesse, leur fiabilité et leur précision. La robustesse est proche de la fiabilité mais un capteur fiable peut donner des signaux réguliers et à un niveau reflétant bien la grandeur physique qui le fait agir et tomber en panne à la moindre chute de température ou après quelques heures de fonctionnement. A l'inverse, un capteur robuste et résistant à toute épreuve peut donner de temps en temps des valeurs fantaisistes et se laisser divertir par des perturbations de son environnement. La précision est le résultat d'un compromis. Du capteur grand public à l'instrument de mesure de laboratoire, il y a un monde et détecter une présence n'est pas forcément équivalent à mesurer une distance au millimètre près. La précision, comme les deux autres facteurs influence fortement sur le coût de ces équipements et sur l'effort pour les mettre en œuvre de façon satisfaisante. Il est en effet question d'ajustements, de calibrage, de calcul et de correction d'erreurs qu'un circuit programmable comme l'Arduino peut correctement gérer mais qui ne pourra pas donner une précision meilleure que celle du plus faible de ses capteurs.

Mais dans le cadre de ce workshop, les préoccupations seront plus basiques puisque nous nous limiterons à une mesure approchée de quelques grandeurs physiques.

2. Une démonstration

Une démonstration va permettre de se familiariser avec 3 capteurs : un capteur de température, un capteur de lumière visible et un capteur de rayonnement infrarouge. Pour illustrer, chaque capteur est couplé avec un actionneur.

Il ne reste qu'à déterminer le comportement attendu de l'Arduino pour écrire l'algorithme puis le code correspondant. Pour rester pragmatique et laisser entrevoir des possibilités d'application, un besoin est exprimé pour chaque cas.

| Besoin | Capteur | Que fait l'Arduino ? | Actionneur |
|---|---|---|--|
| Lorsque la température dépasse 30°C, un ventilateur doit se mettre en marche. Sa vitesse est constante. | Un capteur de température de type TMP01 convertit la température en tension | Il mesure à chaque boucle la tension aux bornes du capteur et la convertit en degrés Celsius. Puis il compare cette valeur avec le seuil de déclenchement du moteur. Si elle est supérieur il alimente la borne plus du moteur | Un moteur électrique à vitesse fixe se met en marche tant que la température est au-dessus du seuil |
| Lorsque la luminosité augmente et dépasse un certain seuil, des volets occultants se déploient pour réduire la luminosité | Une photorésistance voit sa résistance varier en fonction de la luminosité. Un montage en pont diviseur permet de traduire cette résistance en tension | Il mesure la tension en sortie du pont diviseur et la compare et la convertit en une valeur d'angle qui est passée à un servomoteur. Plus la luminosité, donc la tension, est élevée et plus l'angle est grand sans dépasser 180° | Le bras d'un servomoteur sur lequel est accroché un rideau occultant se déplace à la valeur d'angle correspondant à l'ensoleillement |
| Une alarme doit retentir lorsque la distance entre un capteur et un objet est inférieure à 20cm | Un capteur de rayons infrarouges est couplé avec un émetteur et la différence de durée entre émission et réception d'un signal est convertie en tension | Il convertit la tension en distance si le circuit du capteur ne le fait pas. Lorsque la distance est inférieure au seuil de 20cm, il envoie un signal de fréquence 440Hz et d'une durée définie vers la borne + d'un petit haut-parleur | Un bip à 440hz est émis par un petit haut-parleur (buzzer) |

Chacun de ces capteurs partage les mêmes caractéristiques :

- Il fait partie d'un montage intégrant d'autres composants électroniques qui sont nécessaires à son fonctionnement et/ou qui l'améliorent (filtrages anti-parasites) ;
- Ils sont reliés aux entrées analogiques de l'Arduino qui convertissent la tension en niveaux de sortie selon une résolution qui dépend de la fréquence du processeur (conversion A-D ou analogique en digital) ;
- Le niveau de tension mesuré est utilisé soit directement soit fait l'objet d'un calcul. Pour atténuer les erreurs de mesure, un traitement particulier peut être fait : suppression de valeurs aberrantes, calcul d'une moyenne ;

- Lorsque le code de l'Arduino positionne un actionneur en fonction de la valeur d'un capteur au cours d'une boucle, l'actionneur reste dans le même état pendant les boucles suivantes. Il est donc nécessaire de tester et d'attribuer également la valeur de repos de l'actionneur lors de chaque boucle.

Il existe de nombreux capteurs utilisables pour compléter cet exemple : capteurs de couleurs (en fait, plusieurs capteurs de lumières ayant des filtres différents), de pression, de déplacement (accéléromètre), de son voire d'ultra-sons, de fumée, d'humidité...

3. Un cas pratique d'utilisation d'un capteur : la mesure de distance

Il existe plusieurs capteurs pour mesurer une distance : lasers, infrarouge, ultrasons. Les deux derniers types sont les plus répandus en robotique de loisirs. Le cas pratique mettra en œuvre un couple émetteur-récepteur à ultrasons, sur le principe du sonar.

3.1. Un peu de technologie

La mesure de distance est le domaine de la télémétrie. Ce nom provient du grec ancien tele qui veut dire loin et de mesure. La télémétrie est l'art de mesurer les distances et le télémètre est l'appareil qui permet de le faire.

Un télémètre permet, par exemple, de mesurer la distance entre le robot et une cible pour évaluer la distance de tir. Il est basé sur le principe du sonar.

Qu'est ce qu'un sonar? Ce n'est pas un radar même s'il partage le même principe : celui de la mesure du temps entre l'émission d'une onde et sa réception lorsqu'elle rencontre un obstacle. Un radar utilise des ondes électromagnétiques et permet une meilleure détection mais est plus compliqué à mettre en œuvre. Un sonar utilise les ultrasons, non audibles, qui se propagent différemment selon le milieu. Il fonctionne mieux dans un milieu homogène et c'est pourquoi il est utilisé pour détecter et situer les objets sous l'eau.

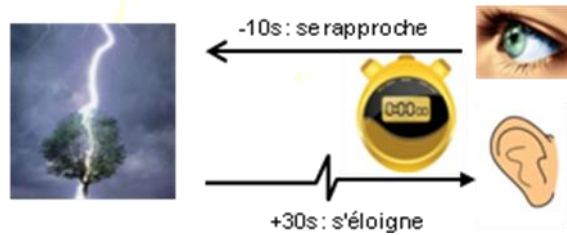
La vitesse du son permet de calculer la distance parcourue par un son. Elle varie selon les différents milieux. Dans l'air et au niveau de la mer, elle est approximativement de 343 mètres par seconde, c'est-à-dire qu'en une seconde, le son parcourt 343 mètres.



C'est plus lent que la vitesse de la lumière et c'est pourquoi quand on aperçoit un éclair, on attend plus ou moins longtemps pour entendre le grondement du tonnerre. Par exemple, si on compte 3 secondes entre l'éclair et le tonnerre alors la foudre est tombée à 3×343 soit 1029 mètres, soit un peu plus de 1 kilomètre.



En examinant la variation de la durée, on peut savoir si l'orage s'éloigne ou s'il se rapproche.



En résumé, la vitesse du son est utile pour mesurer une distance à l'aide d'un sonar

3.2. Application à la robotique

Pour les Trophées 2014, JAMK Robotique a utilisé un sonar pour calculer la distance entre le robot et sa cible :

L'onde sonore

1 Elle est envoyée par l'émetteur du sonar pendant un très court instant, on parle de dizaines de microsecondes, soit un temps que l'être humain ne peut pas contrôler mais qu'un Arduino sait gérer.

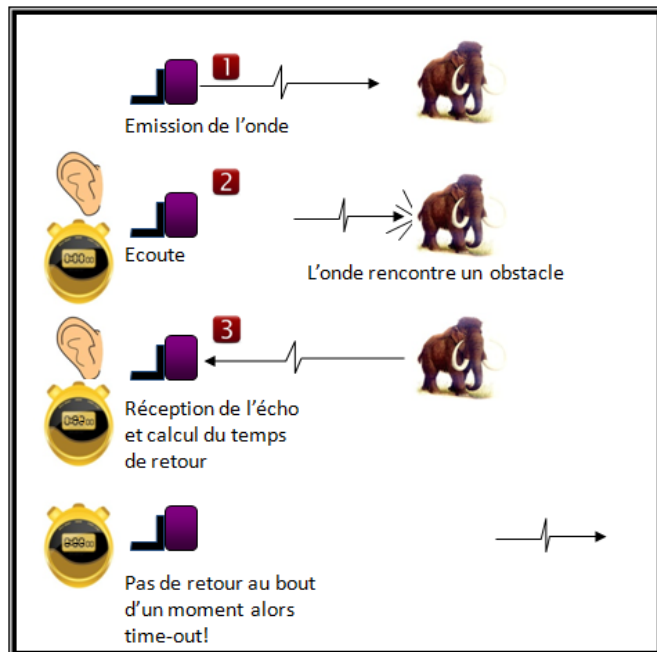
2 Ensuite, le capteur du sonar écoute en attendant que l'onde revienne si elle est réfléchiée par un obstacle;

3 Si c'est le cas, le capteur nous donne le temps écoulé entre l'émission et la réception. En fait, il faut aller lire cette information dans son circuit électronique avec un Arduino mais ce n'est pas très compliqué.

Si l'onde n'a pas rencontré d'obstacle alors au bout d'un moment, le récepteur dira qu'il n'a rien entendu et dira juste « time-out! », c'est-à-dire « temps écoulé » et l'Arduino lira qu'il n'y a pas d'écho.

Le principe est assez simple et le résultat dépend de la précision et de la fiabilité du Sonar.

Nous utiliserons un modèle bon marché donc pas forcément très précis



L'Arduino donne le top pour l'émission de l'onde, il attend la réponse et il va lire la durée enregistrée dans le circuit du sonar. Ensuite, il calcule la distance à partir de la durée d'aller et retour qui vient d'être mesurée et de la vitesse du son.

Un exemple de sonar : le SRF-05



Pour un peu moins d'une vingtaine d'euros, ce petit module qui comprend un émetteur et un récepteur est très utilisé.

C'est un bon rapport qualité/prix et il existe d'autres modules plus chers qui ont un seul composant émetteur-récepteur et qui font également le calcul de distance.

Celui-ci donne des impulsions dont la largeur est comprise entre 100 microsecondes et 18 millisecondes et peut faire des mesures correctes sur **une distance comprise entre 3 cm et 4 mètres**.

Il faut attendre 50ms entre 2 impulsions pour éviter un faux écho.

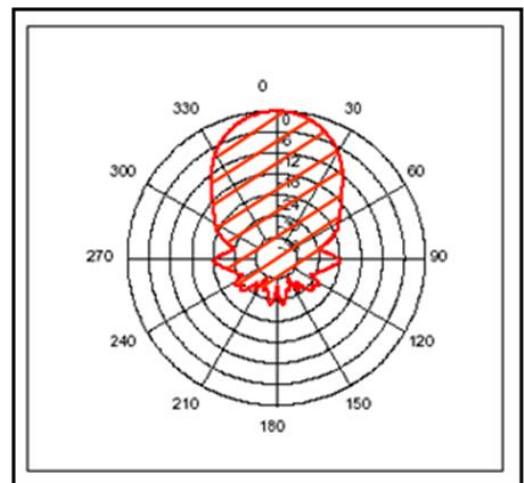
Sur ce modèle, on peut utiliser une ou deux bornes du circuit pour piloter le sonar et recueillir ses données. Il y a aussi 2 bornes qui permettent de l'alimenter en 5V mais il fonctionne aussi en 3,3V (tension utilisée par l'Arduino Due)

L'étalonnage

Au démarrage de l'Arduino, le circuit règle sa mesure de distance en évaluant ce qui se trouve droit devant lui. Si un obstacle est présent à cet instant, la mesure peut être faussée. Il faut veiller à ce que le champ soit dégagé.

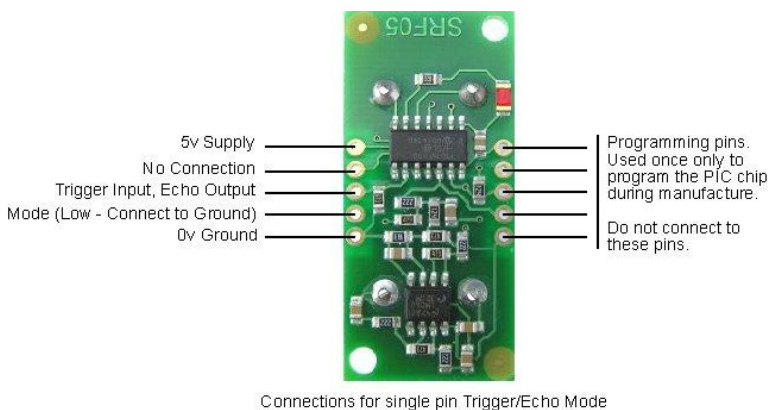
La zone de détection

Si le robot est face au 0, alors il pourra détecter les obstacles qui se situent dans la zone rouge hachurée



La connexion avec l'Arduino

2 bornes sont utilisées pour émettre et recueillir le signal même si une seule peut être utilisée avec ce modèle. Les autres bornes ne doivent pas être utilisées



La commande du sonar par l'Arduino est basée sur les principes suivants :

Il faut donc envoyer un signal depuis l'Arduino pour dire au circuit du sonar qu'il doit émettre une onde.

Ca se fait en envoyant un front haut (une impulsion sur la patte de l'Arduino connecté au circuit) pendant 50 microsecondes. **1**

Le circuit envoi ensuite 8 petites impulsions **2**

Puis se met à écouter **3** et attend une réponse pendant 20 millisecondes. Au bout de 30 millisecondes (le fameux « time-out »), il considère que l'écho ne reviendra pas.

Rappel : pour envoyer une impulsion à partir d'une patte de l'Arduino on écrit HIGH avec l'instruction `digitalWrite()`

`digitalWrite(NomDePatte, HIGH);` **1**

Pour lire la durée dans le circuit du sonar, on utilise la fonction `pulseIn`

Durée = `pulseIn(NomDePatte, INPUT);` **3**

Le code du programme permet d'envoyer une onde, de lire la durée puis de calculer et d'afficher le résultat

La gestion des signaux par l'Arduino et le sonar

Le train d'ondes

On peut le visualiser avec un oscilloscope

Le calcul de la distance

Attention, le temps mesuré est celui de l'aller-retour de l'onde sonore, il faut diviser le temps par 2 pour avoir celui d'un aller simple vers l'obstacle!

Pour avoir la distance en mètres, on divise le temps mesuré en secondes par 2 pour avoir le temps de l'aller puis on divise par la vitesse du son (344 m/s) pour avoir la distance.

Donc si on a une durée de 0,002 secondes, soit 2 millisecondes, l'obstacle ou la cible se situe à ...

(il faut $0,002 / 2 = 0,001$ seconde pour l'aller)
 $0,001 \times 344 = 0,344$ mètres soit 34,4 centimètres

IAMK Robotique
Workshop #3

2014

7

A une échelle de temps aussi petite, un oscilloscope est très utile – et même indispensable – pour « voir » le fonctionnement du sonar. La planche ci-dessous illustre ce qui se passe lorsque l'Arduino exécute les instructions de pilotage du sonar.

Comme ce composant a une précision limitée et que l'environnement peut émettre des perturbations, bien que moins importantes que pour un capteur infrarouge, le calcul de la distance est effectué à partir d'une moyenne des lectures sur un intervalle de temps. Une moyenne simple sur une vingtaine de lecture suffit mais les possibilités de calcul de l'Arduino permettent de faire des corrections plus élaborées et de donner une valeur plus précise.

La préoccupation est la même avec un capteur infrarouge, beaucoup plus sensible à une pollution lumineuse. Une solution, basée encore sur la capacité de traitement, consiste à encoder le signal émis et à le décoder par le récepteur. De cette façon, seuls les signaux effectivement émis seront traités par l'Arduino.

7/28

Application avec l'oscilloscope

La gestion des signaux par l'Arduino et le sonar

Signal d'émission Attente de l'écho pendant 25 millisecondes

1 3

Arduino envoie des impulsions vers le circuit

8 petites émissions d'onde sonore

2

Le circuit envoie des ondes sonores devant lui

Code couleur
En bleu : les actions pilotées par l'Arduino
En rouge : les actions du circuit

D'après Devontech

Moniteur série de l'Arduino

Envoyer

pulsetime : 1925

Défilement automatique NL & CR 9600 baud

Programme de test

Arduino Uno en COM5

2.1V
1.1V
0.1V
-0.9V
-1.9V
-2.9V
-3.9V

1 3

Signal d'émission de 50µs

800µs

1,92ms

Départ

Attente de l'écho

Retour de l'écho

1,92ms → 33cm en appliquant la formule de calcul du fabricant (diviser par 58 la durée en microS)

JAMIK Robotique
Workshop #3

2024

8

3.3. Le code de l'Arduino

Il commence, comme toujours, par la déclaration des variables.

```
// --Sonar et variables de calcul
int echoPin = 3;           // echo pin du SRF05 -> ne marche pas en DEFINE donc int
int initPin = 2;          // init pin du SRF05 -> ne marche pas en DEFINE donc int

const int NbMesure = 10;  // Nombre de mesure pour calculer la moyenne
int IdxMesure = 0;        // indice de Mesure
int TotalMesure = 0;      // Total des mesures
int MoyenneMesure = 0;    // Moyenne des mesures = TotalMesure / NbMesure
int MemoMesure = 0;       // Variable de travail
```

Comme la mesure de la distance est utilitaire et sera exécutée régulièrement, nous l'avons décrite dans une fonction qui actualisera la valeur de la variable Distance par une instruction du type :

MesurerDistance (Distance) ;


```

// -- Fonction de mesure
void MesurerDistance(int Distance){
// int MemoMesure = 0; //-- a utiliser si stabilisation par plusieurs cycles successifs
  int TotalMesure = 0;
  int MoyenneMesure = 0;
  int NbLecture = 0;
  Distance = 0;

  for (int IdxMesure = 0; IdxMesure<=NbMesure;IdxMesure++) { // prise de plusieurs mesures puis moyenne
    digitalWrite(initPin, LOW);
    delayMicroseconds(50);
    digitalWrite(initPin, HIGH); // envoi d un signal
    delayMicroseconds(50); // temporisation de 50 microsecondes
    digitalWrite(initPin, LOW); // arret du signal
    Pulsation = pulseIn(echoPin, HIGH); // calcul du temps de retour du signal
    Distance = Pulsation/58; // conversion en centimetre
    TotalMesure = TotalMesure + Distance; // ajout au total
    NbLecture++;
    delay(10);
  }
  MoyenneMesure = TotalMesure/NbLecture; // Calcul de la moyenne
  Distance = MoyenneMesure; // on fait un cast pour ne garder que la partie entier
}

```

La distance mesurée peut être lue sur le moniteur série grâce à l’instruction Serial.print. Mais il est intéressant de l’afficher sur un ensemble de 4 afficheurs à 7 segments de LED

4. Un exemple utile d’actionneur : un afficheur 4 digits à LED

Un afficheur est un exemple d’actionneur puisqu’il exécute les actions prescrites par l’Arduino.

4.1. Principe d’utilisation

Le but est d’utiliser un afficheur qui sera piloté par l’Arduino avec seulement 3 PIN. Un afficheur 4 digits est composé de LED. Le nombre de LEDs varie selon le type d’afficheur et dans l’exemple, il y aura 8 LEDs par afficheur : 7 LEDs qui permettent d’afficher un nombre et une pour le point décimal. L’Arduino devrait donc gérer 32 LEDs avec un nombre de PIN beaucoup plus réduit.

C’est pourquoi deux mécanismes vont simplifier l’utilisation de l’afficheur : un circuit multiplexeur comme le MAX7219 et le protocole SPI (Serial Peripheral Interface) qui va permettre de dialoguer avec lui. Ce protocole ne nécessite que 3 PINs : un pour synchroniser les horloges de l’Arduino (CLOCK ou CLK) et du circuit afin de gérer la transmission de données, 1 pour indiquer l’envoi des données (LOAD ou CS) et un pour transmettre les données (DIN).

L’afficheur nécessite peu de matériel :



Un afficheur 4 digits à 7 segments, bleu Arduino



Un circuit MAX7219, multiplexeur spécialisé dans la gestion d'affichage à LED



Une résistance 10kΩ pour le MAX7219

Coté logiciel, la librairie LedControl va faciliter le code en gérant l'afficheur (activation/désactivation) et en formatant les informations qui pourront prendre la forme de chiffres ou de caractères en sélectionnant les LEDs adéquates. Le principe est applicable aussi à une matrice de LEDs.

La librairie doit être déclarée par une clause include en début du code.

```
#include <LedControl.h>
```

Le principe de fonctionnement d'un afficheur avec l'Arduino est le suivant :

Arduino et les afficheurs : comment ça marche?

Le principe

Un programme dans l'Arduino donne les instruction d'affichage de nombres, de lettres, ou de motifs

Un circuit (MAX7219) traduit les instructions de l'Arduino en signaux pour l'afficheur

L'afficheur se contente d'éclairer des LEDs selon les signaux du circuit

Le circuit MAX7219 permet de gérer simplement jusqu'à 8 afficheurs via la library LEDControl de l'Arduino

L'Arduino communique avec l'afficheur avec seulement 3 PINs

Notez bien : il est possible de gérer chaque LED de l'afficheur directement par l'Arduino mais comme 1 PIN = 1 LED et que 1 afficheur = 8 LEDs, il faudrait 32 PINs pour commander le tout!

Disposées en matrices les LED peuvent aussi afficher des caractères ou des graphiques

Source : http://www.pjrc.com/teensytd_libs_Matrix.html

Vue sur l'afficheur et le circuit MAX7219 une fois câblés

La plaque d'essai est indispensable pour tester et mettre au point le montage

JAMK Robotique Workshop #3

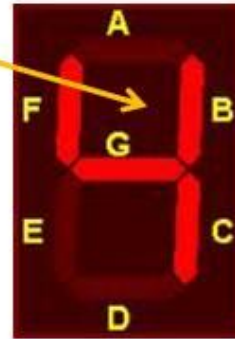
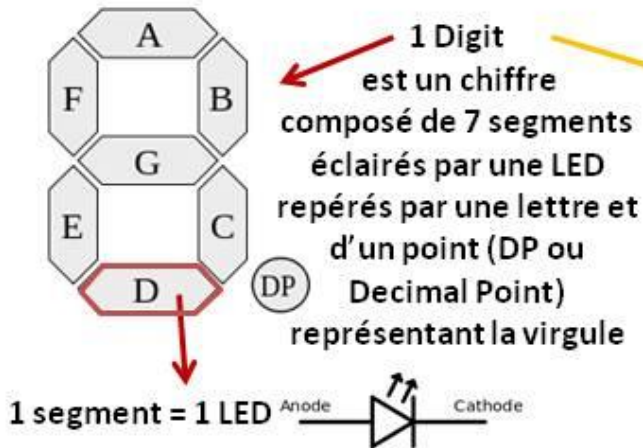
2014

1
0

4.2. Pas à pas de montage d'un afficheur

Pas à pas de montage – Détail

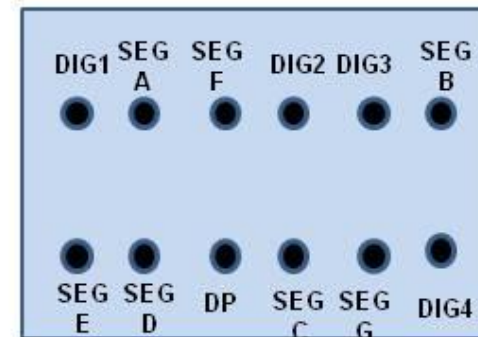
1 – Le module afficheur LED à 4 Digits – 7 segments



En éclairant plusieurs LEDs ou segments, on compose des chiffres ou des lettres (certaines lettres seulement... mais il suffit d'afficher les segments un par un pour créer d'autres caractères)



Les afficheurs sont composés de plusieurs digits assemblés avec une anode ou une cathode commune




Les 12 pattes de l'afficheur sont connectées aux 7 segments (SEG 1 à 7) + le point décimal (DP) et à chaque digit (DIG 1 à 4)

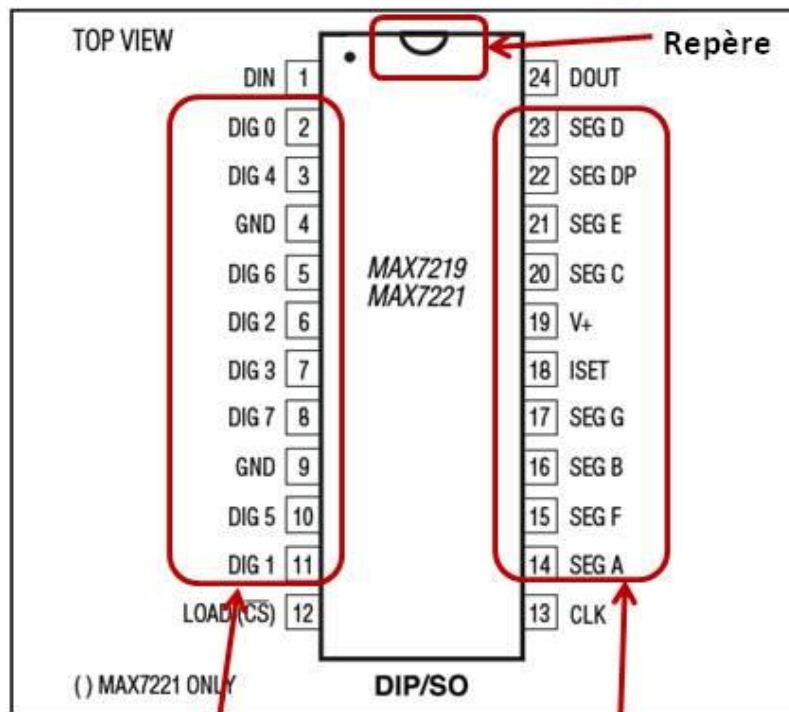


Les LED peuvent aussi être disposées en matrices et afficher des caractères ou des graphiques

Pas à pas de montage – Détail

2 – Le circuit MAX7219  est un multiplexeur spécialisé dans la gestion d'affichage à LED

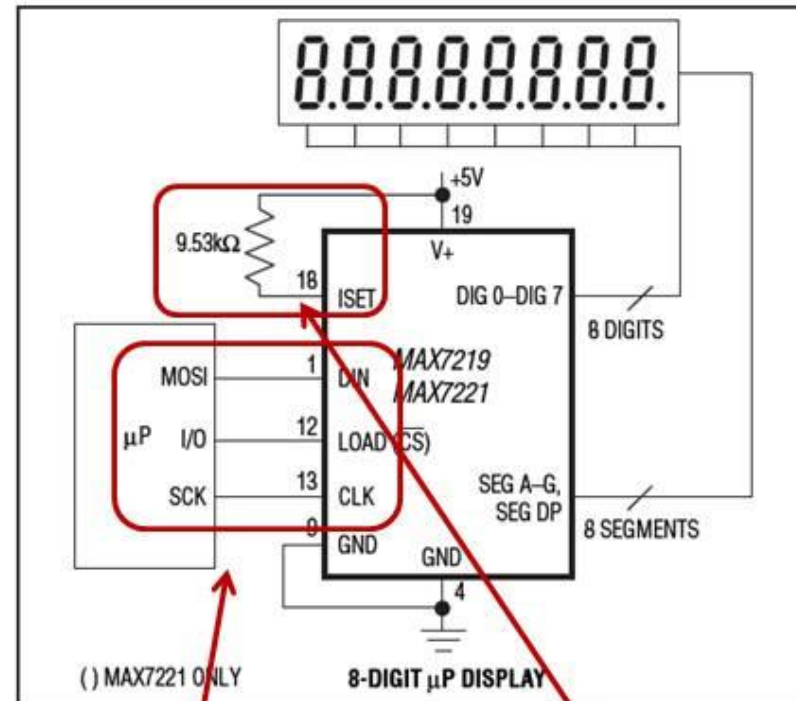
Pin Configuration



Les pattes DIG correspondent à chaque afficheur 7 segments

Les pattes SEG correspondent à chaque segment d'un afficheur

Typical Application Circuit



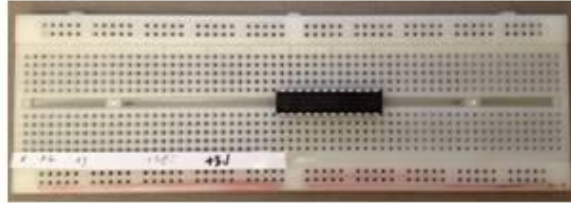
Les pattes DIN, LOAD et CLK sont connectées aux PINs de l'Arduino et gérées par la library LEDControl

La résistance de 10k Ω limite le courant dans le circuit et l'afficheur

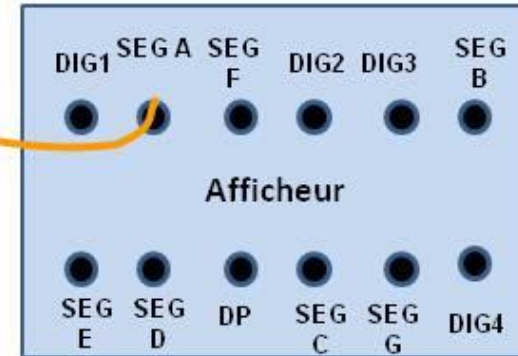
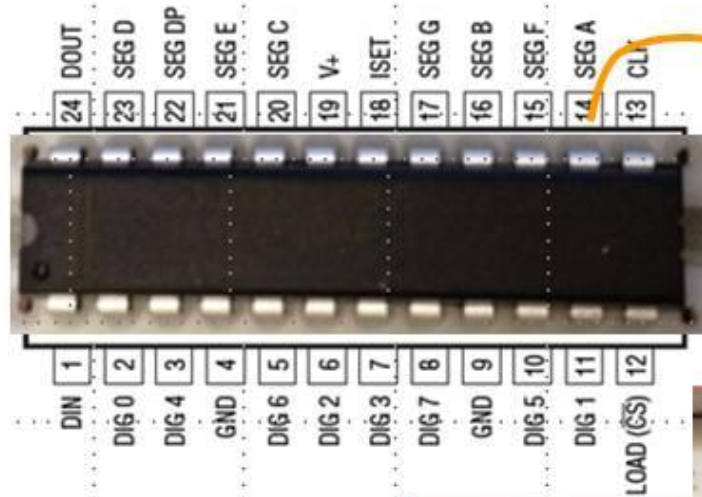
Pas à pas de montage – Détail

1

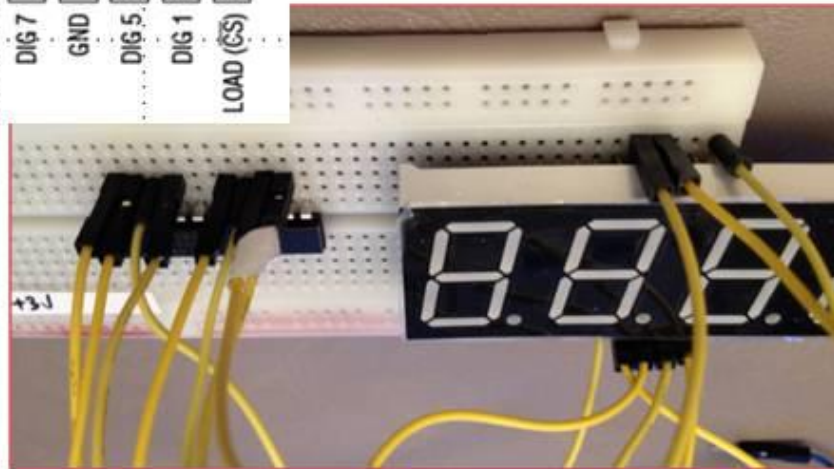
Implanter
(avec précaution)
le circuit MAX7219 et
l'afficheur sur la plaque
d'essai, à cheval sur le rail
central



2



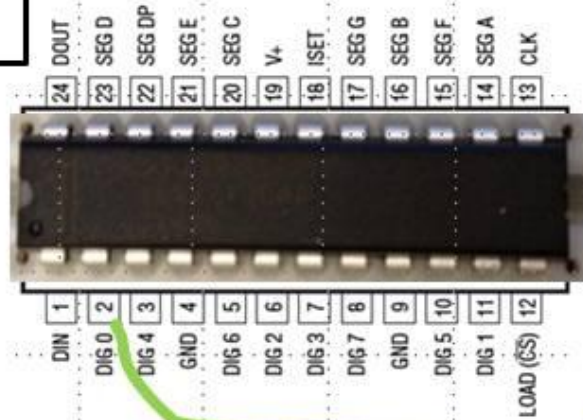
Connecter les pattes SEG de
l'afficheur aux pattes SEG du
circuit MAX7219



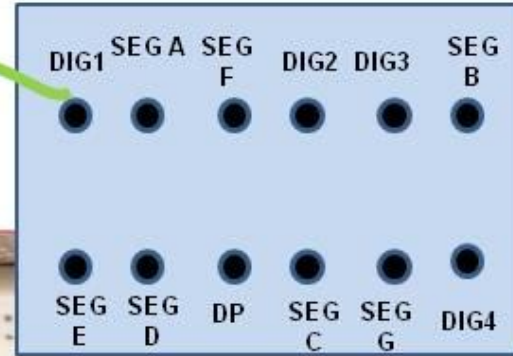
Vérifier le câblage
à chaque étape
pour éviter des
erreurs difficiles
à détecter quand
tous les fils sont
connectés

Pas à pas de montage – Détail

3



Vérifier le câblage avant l'étape suivante

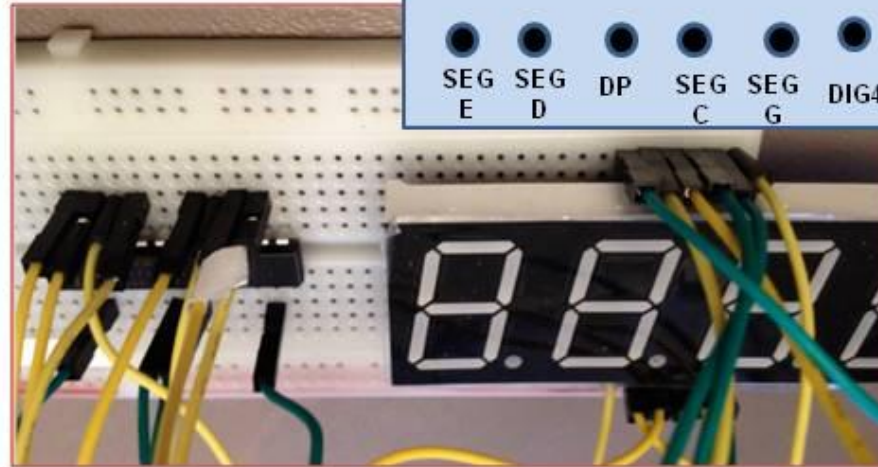


Connecter les pattes DIG de l'afficheur aux pattes SEG du circuit MAX7219

DIG 0 — DIG 1

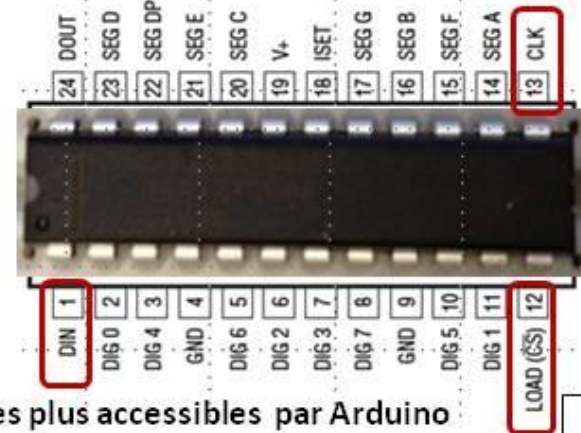
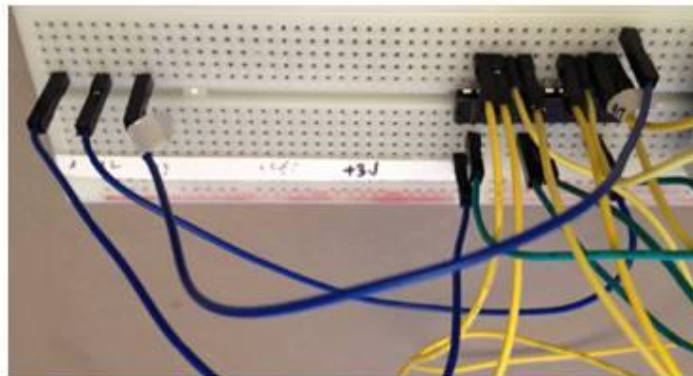


les digits sont numérotés à partir de 0 sur le circuit MAIS à partir de 1 sur l'afficheur



4

La câblage de l'afficheur est terminé
On connecte le circuit à l'Arduino



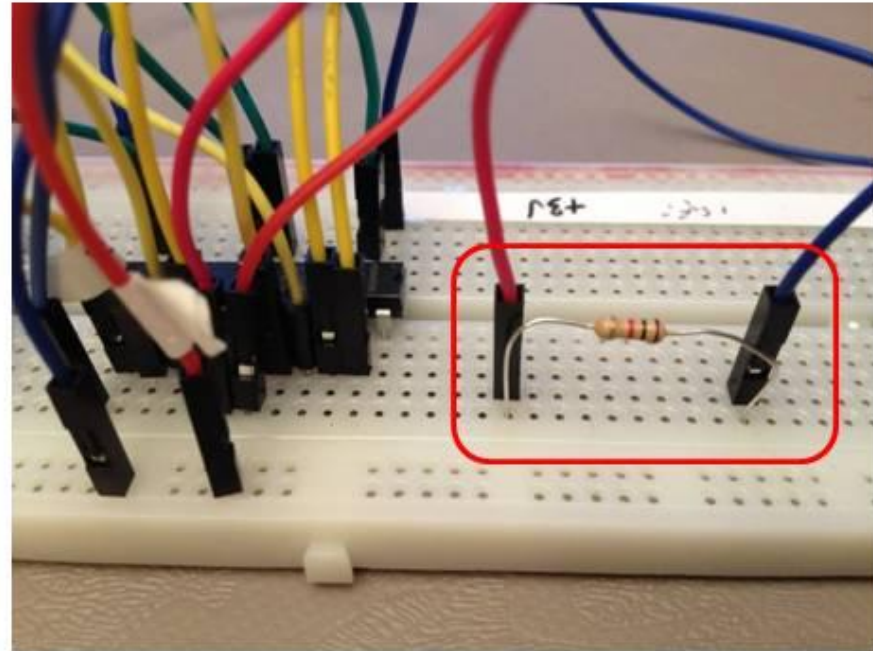
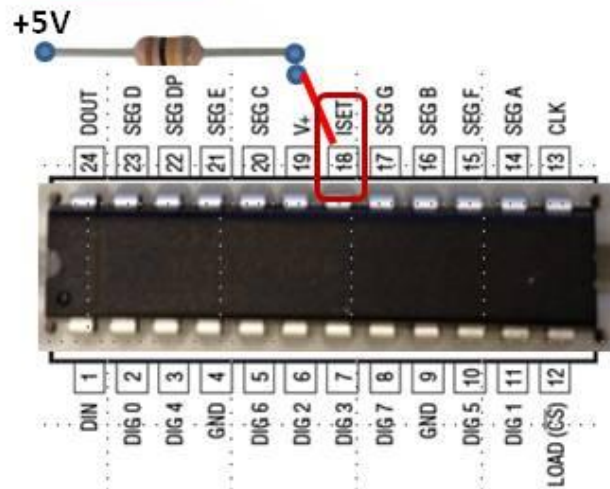
Connecter les pattes CLK, DIN et LOAD à 3 rangées plus accessibles par Arduino

Pas à pas de montage – Détail

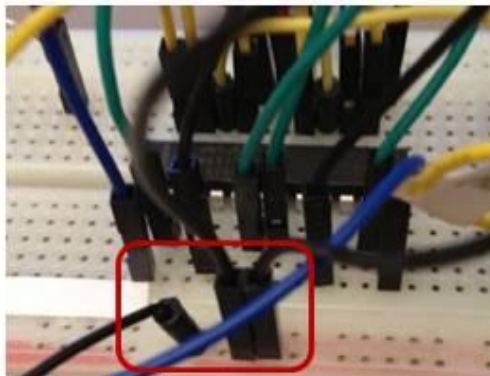
5



La patte ISET du circuit doit être reliée à l'alimentation +5V via une résistance de 10kΩ pour limiter la consommation de l'afficheur



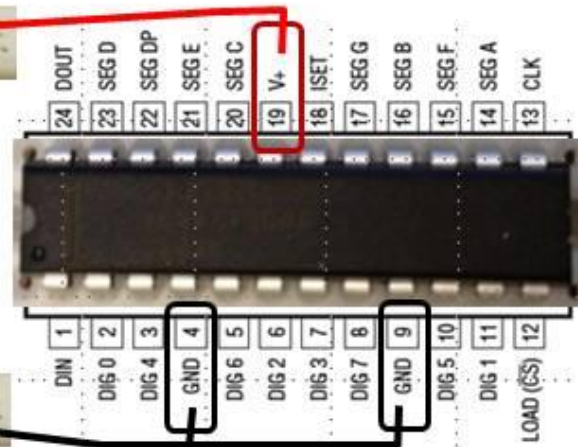
6



+5V

Connecter les 2 pattes GND à une ligne GND sur la plaque d'essai et la patte V+ à l'autre ligne

GND



Pas à pas de montage – Détail

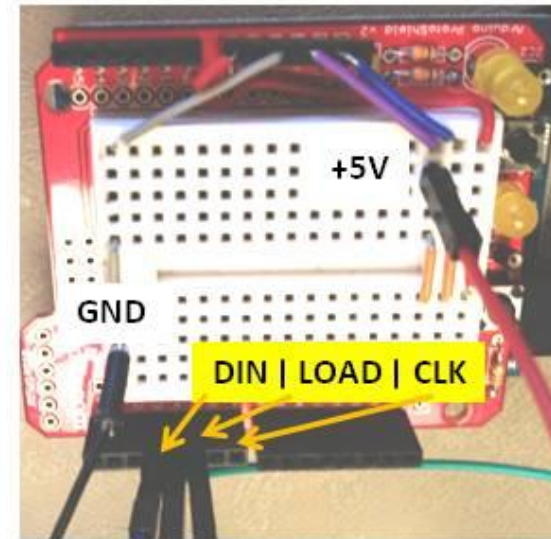
7

Connecter l'Arduino à la plaque d'essai.
L'Arduino assure l'alimentation +5V et la masse (GND).
On connecte donc les PINs +5V et GND sur les rails correspondants

Enfin, on connecte les PINs de l'Arduino aux pattes DIN, CLK et LOAD du circuit à partir de la plaque d'essai



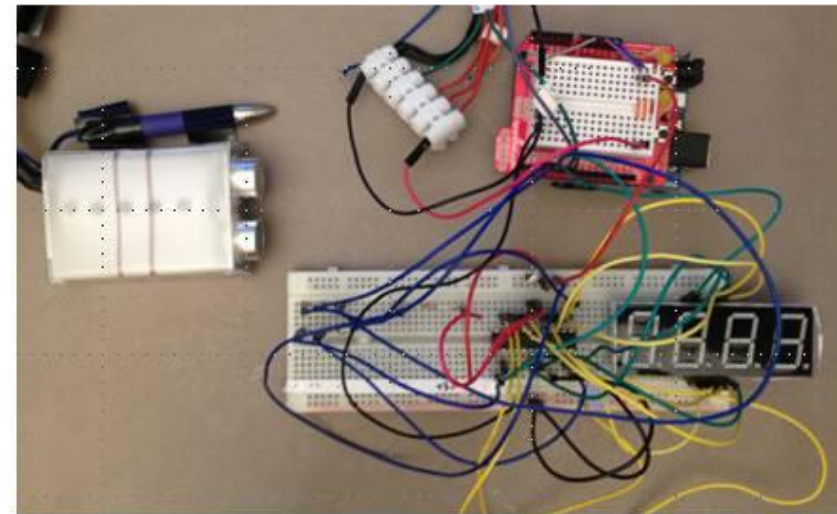
Vérifier le câblage une dernière fois avant de mettre sous tension



8

Le câblage de l'afficheur est terminé, il ne reste plus qu'à le tester à l'aide de programmes de l'Arduino

- 1 - Charger le programme `TestAfficheurLEDV2.ino`
- 2 - Le résultat doit être l'affichage des chiffres 1 à 8 successivement de gauche à droite
- 3 - Si les DIGITS ne s'allument pas dans le bon ordre, vérifier le câblage des broches DIG
- 4 - Si les chiffres ne s'affichent pas correctement, alors vérifier le câblage des broches SEG



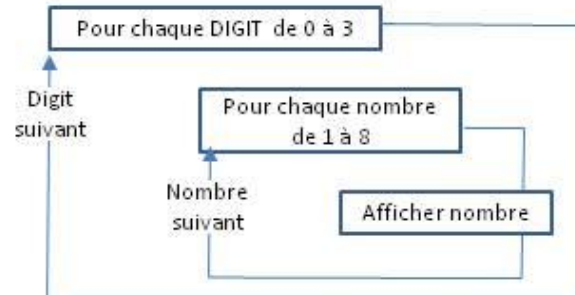
Pas à pas de montage – Le programme de test

Déclaration des variables

Ces 3 portions de code se retrouvent dans tous les programmes gérant un afficheur avec la library LEDControl

Initialisation
Setup()

Boucle principale
loop()



```
TestAfficheurLEDV2 | Arduino 1.0.5
Fichier Édition Croquis Outils Aide

TestAfficheurLEDV2
// JAMK Robotique - 26 dec 2013
// Test afficheur LED

#include <LedControl.h> // Gere la communication SPI avec le circuit
#define DIN 2           // Digital IN
#define CLK 4           // Clock
#define LOAD 3          // Load
#define NumChips 1     // on utilise un seul circuit MAX7219

float nombre = 0;      // Le nombre a afficher sur chaque digit

LedControl lc=LedControl(DIN,CLK,LOAD,NumChips); // creation de l'objet afficheur

void setup(){
  // init du circuit MAX7219, réglage de l'intensite et mise a zero
  // le circuit 1 a comme adresse 0
  lc.shutdown(0,false); // reveille le circuit qui est en veille a l'init
  lc.setIntensity(0,8); // entre 1 et 15, 8 est une bon compromis
  lc.clearDisplay(0); // efface l'affichage
  Serial.begin(9600); // pour ecrire en mode ligne sur le moniteur serie
}

void loop(){
  // -- Test d'affichage digit par digit : on affiche de 8 a 0 sur chaque afficheur.
  // -- pour verifier que les broches DIG sont dans le bon ordre
  // ATTENTION les 4 afficheurs sont numerotes de 0 à 3 0 - 1 - 2 - 3
  for(int i = 0; i<4; i++){ // digit 0 pour le circuit MAX7219 et 1 pour l'afficheur
    lc.clearDisplay(0);
    for (nombre = 1; nombre < 9; nombre++){
      Serial.print("Afficheur : ");
      Serial.println(i);
      Serial.print("valeur : ");
      Serial.println(nombre);
      lc.setDigit(0,i,nombre,false);
      delay(200);
    }
    // -- Le resultat doit etre l'affichage des chiffres 1 a 8 successivement de gauche
    delay(200);
  }
}
```

Affichage sur le moniteur série

```

void afficheValeur(int v){
  //-- La valeur v est décomposée en centaines, dizaines et unités
  int unites;
  int dizaines;
  int centaines;
  boolean negatif = false;
  //-- Contrôle des limites et du signe (negatif gère par affichage)
  if(v < -999 || v > 999)
    return;
  if(v<0) {
    negatif=true;
    v=v*-1;
  }
  unites=v%10;
  v=v/10;
  dizaines=v%10;
  v=v/10;
  centaines=v;
  lc.clearDisplay(0);
  if(negatif) {
    lc.setChar(0,0,'-',false);
  }
  else {
    lc.setChar(0,0,' ',false);
  }
  //-- Ecriture de la valeur chiffre par chiffre
  lc.setDigit(0,1,(byte)centaines,false);
  lc.setDigit(0,2,(byte)dizaines,false);
  lc.setDigit(0,3,(byte)unites,false);
}

```

```

void afficheValeur(int v){
  //-- Affiche une valeur v entre -999 et 999 sr 4 digits
  //-- Cette fonction vient du site http://playground.arduino.cc/Main/LedControl

  //-- La valeur v est décomposée en centaines, dizaines et unités

  int unites;
  int dizaines;
  int centaines;
  boolean negatif = false;

  Serial.println(v);

  //-- Contrôle des limites et du signe (negatif gère par affichage)
  if(v < -999 || v > 999)
    return;
  if(v<0) {
    negatif=true;
    v=v*-1;
  }
  unites=v%10;
  v=v/10;
  dizaines=v%10;
  v=v/10;
  centaines=v;
  lc.clearDisplay(0);
  if(negatif) {
    //-- Affichage du caractère '-' dans la première colonne
    lc.setChar(0,0,'-',false);
  }
  else {
    //-- première colonne à blanc
    lc.setChar(0,0,' ',false);
  }
  //-- Ecriture de la valeur chiffre par chiffre
  lc.setDigit(0,1,(byte)centaines,false);
  lc.setDigit(0,2,(byte)dizaines,false);
  lc.setDigit(0,3,(byte)unites,false);
}
  //-- Fin de fonction afficheValeur

```

Zoom sur la fonction d'affichage d'une valeur

Comment afficher autre chose que des chiffres?

En standard, les chiffres de 0 à 10 et quelques lettres peuvent être affichés directement par la fonction `setChar()`

```
-- Afficher un caractere sur un afficheur 7 segments
'0','1','2','3','4','5','6','7','8','9','0',
'A','b','c','d','E','F','H','L','P',
'.','-','_',''
```

```
void setChar(int addr, int digit, char valeur, boolean dp);
```

addr numero de l afficheur
digit position du caractere sur l afficheur
valeur caractere a afficher
dp point decimal

```
lc.setChar(0,0,'1',false);
```

MAIS la fonction `setRow()` permet de commander chaque LED de chaque afficheur à l'aide d'un mot binaire mettant en correspondance chaque bit avec un segment

Utiliser des fonctions permet de rendre le code plus facile à lire et à vérifier. Ici : une fonction dédiée à l'affichage de la distance

```
void afficheDist(int dist){
  //-- Affiche le visuel Dist et la valeur de distance

  //-- Affiche le libelle D i s t : valeur 1 = segment allume
  //--                    VABCDEF
  lc.setRow(0,0,B00111101); // D
  lc.setRow(0,1,B00010000); // i
  lc.setRow(0,2,B01011011); // S
  lc.setRow(0,3,B00001111); // t
  delay(delaytime);
  lc.clearDisplay(0);

  //-- Affiche la valeur
  afficheValeur(dist);
  delay(delaytime);
}
//-- Fin de fonction afficheDist
```



5. Application : Coupler un capteur et un actionneur

Un exemple de couplage entre capteur et actionneur consiste à mesurer une distance avec un sonar et à afficher sa valeur sur un afficheur 4 digits.

Il suffit de combiner le montage et le code du paragraphe 2 avec ceux du paragraphe 3. La structure du programme est linéaire puisque les deux parties de code s'enchaînent dans l'intérieur de la boucle loop(). Il faudra veiller à ajuster la temporisation par l'instruction delay() pour que l'affichage soit fluide.

Première partie du code : la déclaration des PINs et des variables :

```
1  /*----- SAMEDIS BENEVOLES PLANETTE SCIENCE WORKSHOP#2 - SONAR ET AFFICHAGE -----
2   Dominique MOLLARD, 2 septembre 2015
3   | Algorithmme :
4     1 - Activation du Sonar
5     2 - Mesure
6     3 - Afficher Dist, distance mesurée
7  */
8  /* -----
9  /* -----          DECLARATIONS ET VARIABLES          ----- */
10 /* -----
11 // -- PIN de l arduino - -----
12 // Sonar
13 int echoPin = 9;           // echo pin du SRF05
14 int initPin = 8;          // init pin du SRF05
15
16 // Commande du MAX7129
17 int DIN =                 2;
18 int CLK =                 4;
19 int LOAD =                3;
20
21 // Sonar et variables de calcul
22 const int NbMesure = 10;  // Nombre de mesure pour calculer la moyenne
23 int NbLecture = 0;        // Compteur de lectures pour la boucle
24 int IdxMesure = 0;        // Indice de Mesure pour la boucle
25 int TotalMesure = 0;      // Total des mesures
26 int MoyenneMesure = 0;    // Moyenne des mesures = TotalMesure / NbMesure
27 int MemoMesure = 0;      // Variable de travail
28 unsigned long Pulsation = 0; // durée de l'impulsion
29 unsigned long Distance = 0; // distance calculée à partir de la vitesse du son
30 int Reponse = 0;
```

```

31
32 // -- Afficheur et circuit MAX7219
33 #include <LedControl.h>
34 //-- Librairie optimisee pour une utilisation du MAX7219
35 //-- ATTENTION : lire attentivement le wiki sur son utilisation pour eviter de
36 //-- griller le circuit http://playground.arduino.cc//Main/LedControl
37 //-- Cette library permet d afficher des nombres decimaux ou hexadecimaux, des caracteres
38 //-- alphanumeriques et de piloter chaque segment individuellement. Elle est limitee
39 //-- a 7 digits et utilise l'interface SPI. Pour le Parametrage du MAX7219
40 //-- voir http://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf
41
42 //-- Nombre de MAX7219 utilises : 1 dans le cas present
43 #define NumChips 1
44 #define Intensite 8 // intensite de l'affichage : entre 0 et 15, 8 est un bon compromis
45
46 //-- Creation de l objet LedControl qui communiquera avec le circuit MAX7219
47 LedControl lc=LedControl(DIN,CLK,LOAD,NumChips);
48
49 //-- Delay pour affichage
50 unsigned long delaytime=1000; //-- A ajuster
51
52 //-- Fin declarations et variables -----

```

Le setup() :

```

55 /* ----- */
56 /* ----- SETUP ----- */
57 /* ----- */
58 void setup() {
59 // -- Sonar SRF05
60 pinMode(initPin, OUTPUT); // init pin en OUTPUT
61 pinMode(echoPin, INPUT); // echo pin en INPUT
62
63 //-- init du circuit MAX7219, réglage de l intensite et mise a zero
64 // le circuit 1 a comme adresse 0
65
66 //-- shutdown permet d eteindre l affichage pour economiser la batterie
67 //-- c est le mode natif a l'init de l arduino, donc on reveille l ecran numero 0
68 lc.shutdown(0,false);
69 //-- Réglage de l'intensite entre 0 et 15, 8 est un compromis lisibilite/consom
70 // lc.setIntensity(0,Intensite);
71 lc.clearDisplay(0);
72 //-- Communication avec le PC en mode test
73 Serial.begin(9600);
74 }
75 /* ----- Fin de SETUP ----- */

```

Pour simplifier le code, on a regroupé les instructions d'affichage d'une valeur dans une fonction

```

121 /* -----*/
122 /* ----- FONCTIONS ----- */
123 /* -----*/
124 // -- Fonction d'affichage d'une valeur numérique
125 void afficheValeur(int v){
126     //-- Affiche une valeur v entre -999 et 999 sur 4 digits
127     //-- Cette fonction vient du site http://playground.arduino.cc/Main/LedControl
128     //-- La valeur v est décomposée en centaines, dizaines et unités
129     int unites;
130     int dizaines;
131     int centaines;
132     boolean negatif = false;
133
134     //-- Contrôle des limites et du signe (negatif gère par affichage)
135     if(v < -999 || v > 999)
136         return;
137     if(v<0) {
138         negatif=true;
139         v=v*-1;
140     }

```

```

141     unites=v%10;
142     v=v/10;
143     dizaines=v%10;
144     v=v/10;
145     centaines=v;
146     lc.clearDisplay(0);
147     if(negatif) {
148         //-- Affichage du caractère '-' dans la première colonne
149         lc.setChar(0,0,'-',false);
150     }
151     else {
152         //-- première colonne à blanc
153         lc.setChar(0,0,' ',false);
154     }
155     //-- Écriture de la valeur chiffre par chiffre
156     lc.setDigit(0,1,(byte)centaines,false);
157     lc.setDigit(0,2,(byte)dizaines,false);
158     lc.setDigit(0,3,(byte)unites,false);
159 }
160 //-- Fin de fonction afficheValeur
161
162 /* ----- Fin des fonctions ----- */

```

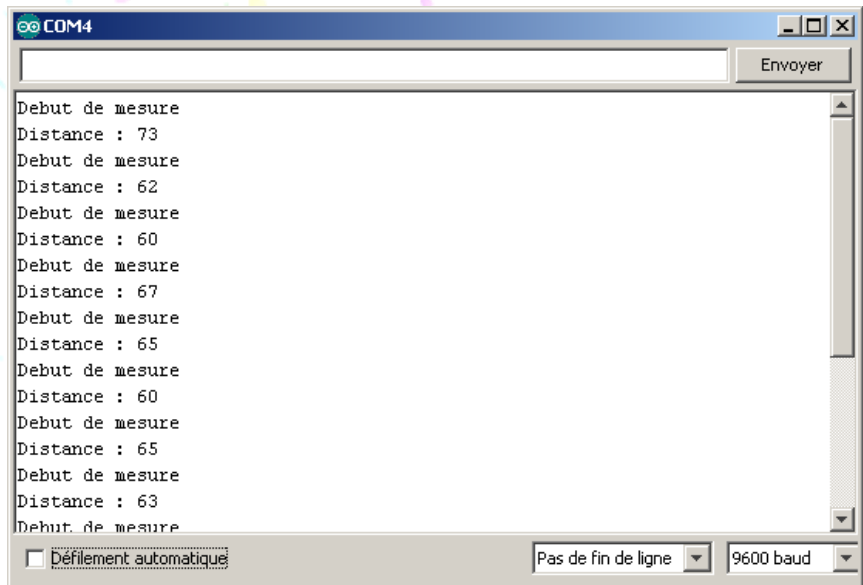
La boucle principale :

```
78 /* ----- */
79 /* -----          BOUCLE PRINCIPALE          ----- */
80 /* ----- */
81
82 void loop() {
83
84     // -- Mesure de la distance
85     TotalMesure = 0;
86     NbLecture = 0;
87     Serial.println("Debut de mesure");
88     for (int IdxMesure = 0; IdxMesure<=NbMesure;IdxMesure++) {
89         digitalWrite(initPin, LOW);
90         delayMicroseconds(50);
91         digitalWrite(initPin, HIGH);           // envoi d un signal
92         delayMicroseconds(50);               // temporisation de 50 microsecs
93         digitalWrite(initPin, LOW);         // arret du signal
94         Pulsation = pulseIn(echoPin, HIGH); // temps de retour du signal
95         Distance = Pulsation/58;            // conversion en centimetres
96         TotalMesure = TotalMesure + Distance; // ajout au total
97         NbLecture++;
98         delay(10);
99     }
100     MoyenneMesure = TotalMesure/NbLecture; // Calcul de la moyenne
101     Distance = MoyenneMesure;             // cast pour avoir la partie entiere
102     Serial.print("Distance : ");
103     Serial.println(Distance);
```

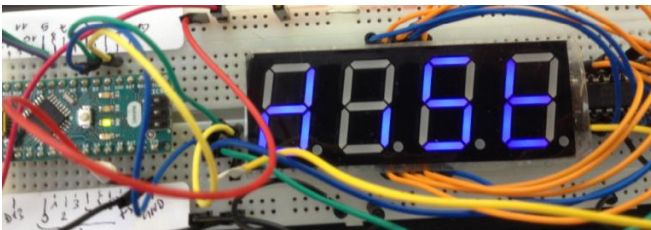
```
105 // -- Affichage de la distance : le visuel Dist puis la valeur de distance
106 //-- Affiche le libelle D i s t : valeur l = segment allume
107 //--          VABCDEF
108 lc.setRow(0,0,B00111101); // D
109 lc.setRow(0,1,B00010000); // i
110 lc.setRow(0,2,B01011011); // S
111 lc.setRow(0,3,B00001111); // t
112 delay(delaytime);
113 lc.clearDisplay(0);
114
115 //-- Affiche la valeur
116 afficheValeur(Distance); // On a cree une fonction affiche valeur pour simplifier
117 delay(delaytime);       // delaytime est à ajuster selon l'affichage souhaite
118 }
119
120 /* -----          Fin de LOOP()          ----- */
```

Le résultat à l'exécution :

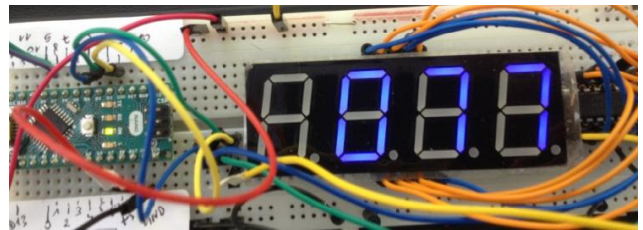
- Sur le moniteur série



- Et sur l'afficheur



Affichage du titre « dist »



Puis de la valeur mesurée, ici 77cm

A noter : le SRF05 a besoin de se calibrer lors de sa mise sous tension ou son activation. Pour obtenir des valeurs précises, il faut mettre un cache sur le sonar lors de l'initialisation.

Le programme associé à cet exemple est **CE_SonarAffichageV2_Final.ino**

6. L'asservissement d'un servomoteur

Nous allons poursuivre l'étude des capteurs et des actionneurs par un cas pratique combinant un capteur à ultra-sons, un afficheur 4 digits et un servomoteur asservi. Il s'agit d'un asservissement très simple en boucle fermée puisque la valeur de l'angle est proportionnelle à la distance mesurée.

En automatique, il existe plusieurs autres types d'asservissement (Intégral ou Dérivé) qui peuvent être mis en œuvre avec un Arduino soit par programme soit en utilisant un circuit calculateur piloté par l'Arduino, pour asservir un robot en vitesse et/ou en position.

Dans cet exemple, nous allons simplement étendre le programme précédent en lui ajoutant un calcul de l'angle en fonction de la distance mesurée, en affichant cet angle puis en le passant à un servomoteur pour qu'il puisse pivoter de la valeur désirée.

Il sera nécessaire d'ajuster les temporisations pour que l'affichage et le déplacement du servomoteur soient fluides.

On ajoute la gestion du servomoteur qui règle l'angle : déclaration du PIN, création de l'objet servo, « attachement » au pin, initialisation.

```
1  /*----- SAMEDIS BENEVOLES PLANETTE SCIENCE WORKSHOP#2 - SONAR, AFFICHAGE + SERVO ----
2  Dominique MOLLARD, 2 septembre 2015
3  Algorithme :
4      1 - Activation du Sonar
5      2 - Mesure
6      3 - Afficher Dist, distance mesuree
7      4 - Calcul de l'angle de tir
8      5 - Rotation du servo
9  */
10 /* -----*
11 /* -----      DECLARATIONS ET VARIABLES      ----- */
12 /* -----*
13 // -- PIN de l arduino - -----
14 // Sonar
15 int echoPin =          9; // echo pin du SRF05
16 int initPin =          8; // init pin du SRF05
17
18 // Commande du MAX7129
19 int DIN =              2; // MAX7129 DIN
20 int CLK =              4; // MAX7129 CLK
21 int LOAD =             3; // MAX7129 LOAD
22
23 // Servomoteur
24 int servoPin =         10; // MonServo
25
```

Les déclarations de variables pour l'afficheur sont identiques.

On ajoute les variables de travail pour le servomoteur.

```

57 // Servomoteurs
58 #include <Servo.h>           // library servo standard
59
60 Servo Lanceur;              // declaration de 1 objet servo
61 int LanceurPos = 0;          // variable de stockage de la position du servo
62 int LanceurAngleCourant = 90; // position courante du servo
63 int LanceurAngleCalcule = 0; // angle cible
64 int LanceurRepos = 0;
65
66 //-- Table de conversion distance/angle a mettre a jour lors des essais
67 int mytable[50]={5,5,5,5,5,5,15,15,15,15,
68     25,25,25,25,25,25,25,25,25,25,
69     25,25,25,52,30,30,30,30,30,30,
70     30,35,35,35,35,40,45,45,45,45,
71     50,50,50,55,55,55,60,60,60,60};

```

La table de conversion permet de trouver la valeur de l'angle correspondant à une distance. Cette valeur peut être calculée à l'aide d'une équation parabolique ou mesurée lors d'essais. Pour l'utiliser, le programme divise la distance par 10, ne retient que la partie entière et utilise la valeur résultante comme index pour accéder à une valeur de la table. Par exemple, pour une distance de 150cm, l'angle correspondant est mytable[150/10], soit mytable[15] = 25°.

Le servomoteur est initialisé dans le setup() :

```

94 //-- Servo lanceur : initialisation et mise en position repos
95 Lanceur.attach(servoPin);
96 pinMode(servoPin, OUTPUT);
97 Lanceur.write(LanceurRepos);

```

Dans la boucle loop(), on ajoute la gestion de l'angle après l'affichage de la distance :

```

139 //-- Affiche la valeur de la distance
140 afficheValeur(Distance); // On a cree une fonction affiche valeur pour simplifier
141 delay(delaytime);       // delaytime est à ajuster selon l'affichage souhaite
142
143 //-- Calcul de 1 angle
144 LanceurAngleCalcule = CalculerAngle(Distance);
145
146 //-- Affiche la valeur de 1 angle
147 afficheAngl(LanceurAngleCalcule); // On a cree une fonction affiche valeur pour simplifier
148 delay(delaytime);           // delaytime est à ajuster selon l'affichage souhaite
149
150 //-- Rotation du servo lanceur par un déplacement progressif
151 LanceurPos = LanceurAngleCalcule; // Voir si AngleCalcul est en doublon avec LanceurPos
152 LanceurAngleCourant = Lanceur.read(); // on lit la valeur courante de 1 angle
153 Lanceur.write(LanceurAngleCalcule);
154 delay(30);
155
156 AffichePret();
157 delay(1000);
158
159 //-- Simulation de tir, dans la pratique on actionnerait un lanceur de qq chose
160 AfficheTir();
161 delay(3000);

```

afficheAngl(), AffichePret() et AfficheTir() sont des fonctions déclarées plus loin et qui permettent de simplifier le code principal en isolant leur code et en facilitant sa réutilisation. A noter que afficheAngl(paramètre) est une fonction qui utilise la variable LanceurAngleCalcule comme paramètre et que la fonction CalculerAngle(paramètre) renvoie une valeur.

```
208 void afficheAngl(int angl){
209     //-- Affiche le visuel Angl et la valeur de l angle
210
211     //-- Affiche le libelle AngL : valeur l = segment allume
212     //--          VABCDEFG
213     lc.setRow(0,0,B01110111); // A
214     lc.setRow(0,1,B00010101); // n
215     lc.setRow(0,2,B01111011); // g
216     lc.setRow(0,3,B00001110); // L
217     delay(delaytime);
218     lc.clearDisplay(0);
219     //-- Affiche la valeur
220     afficheValeur(angl);
221     delay(delaytime);
222 }
223 //-- Fin de fonction afficheAngl
224
225 // -- Fonction de calcul de l angle
226 int CalculerAngle(int dist){
227     int angl = 0;
228     int indice = 1;
229     //-- en entree : distance mesuree par le sonar
230     //-- en sortie : angle en degres
231     //-- On divise dist par 10 pour avoir l indice du tableau
232     indice = dist/10;
233     if ((indice > 0) && (indice < 50)){
234         angl = mytable[indice-1]; //-- A TESTER avec toutes les valeurs de distance
235     }
236     return angl;
237 }
```

Les deux dernières fonctions permettent d'afficher un libellé sur l'afficheur à LED pour faciliter la compréhension des valeurs :

```

239  //-- Fonction d'affichage du libelle Pret
240  void AffichePret(){
241      //-- Affiche le visuel Pret
242      //--          VABCDEFG
243      lc.setRow(0,0,B01100111); // P
244      lc.setRow(0,1,B00000101); // r
245      lc.setRow(0,2,B01001111); // E
246      lc.setRow(0,3,B00001111); // t
247      delay(delaytime);
248      lc.clearDisplay(0);
249  }
250  //-- Fin de fonction affichePret
251
252  //-- Fonction d'affichage du libelle tir
253  void AfficheTir(){
254      //-- Affiche le visuel Pret
255      //--          VABCDEFG
256      lc.setRow(0,0,B00000000); // espace
257      lc.setRow(0,1,B00001111); // t
258      lc.setRow(0,2,B00010000); // i
259      lc.setRow(0,3,B00000101); // r
260      delay(delaytime);
261      lc.clearDisplay(0);
262  }
263  //-- Fin de fonction afficheTir

```

Ce programme (CE_SonarAffichageServoV2_Final.ino) peut bien évidemment être complété pour insérer une séquence de tir avec un bouton-poussoir déclenchant la mesure, l'affichage, la rotation du servomoteur et le lancer.

*

* *