Mis à jour le
30 aout 2016

GANTSONAR V.2.1

Dispositif haptique portatif pour la perception de l'environnement par des mal-voyants

Yves Le Chevalier MyHumanKit aout 2016

NB: la version 2.1 diffère de la V1 uniquement par la façon dont le programme fonctionne, le circuit électronique et ses composants étant absolument identiques.

Ce système est librement inspiré des travaux de l'américain Steve Hoefer.
Il consiste en un dispositif à fixer sur le dos de la main et qui sonde en permanence l'environnement grâce à deux sonars à ultrasons orientés à 90 degrés l'un de l'autre et fonctionnant de façon indépendante. Si un obstacle est détecté par l'un des sonars, sa présence se traduit par une vibration du même côté que celui du sonar. La force de cette vibration est inversement proportionnelle à la distance mesurée. Les deux sonars peuvent détecter des obstacles à des distances différentes et provoquer ainsi des vibrations différentes de chaque côté de la main.

La perception de l'environnement se fait donc en interprêtant les variations de vibrations que l'on ressent en balayant l'espace de la main et, bien entendu, un certain apprentissage est nécessaire pour utiliser ce système de façon naturelle.

Du fait de la position en angle droit des deux sonars, il y a un espace non couvert entre les deux champs de détection. Cet angle mort est important puisqu'il permet de sentir de facon précise la position d'un objet lors de l'arrêt momentané des vibrations au cours d'un balayage continu de la main. Cet objet se trouve alors à ce moment là, exactement dans l'axe de la main.

A noter que, s'il est possible de détecter aussi bien des objets ou des obstacles dans un plan horizontal (il suffit pour cela de balayer l'espace de droite à gauche avec la main à plat), il est possible de le faire aussi dans un plan vertical en balayant l'espace devant soi de bas en haut en tournant la main de 90 degrés sur le côté.

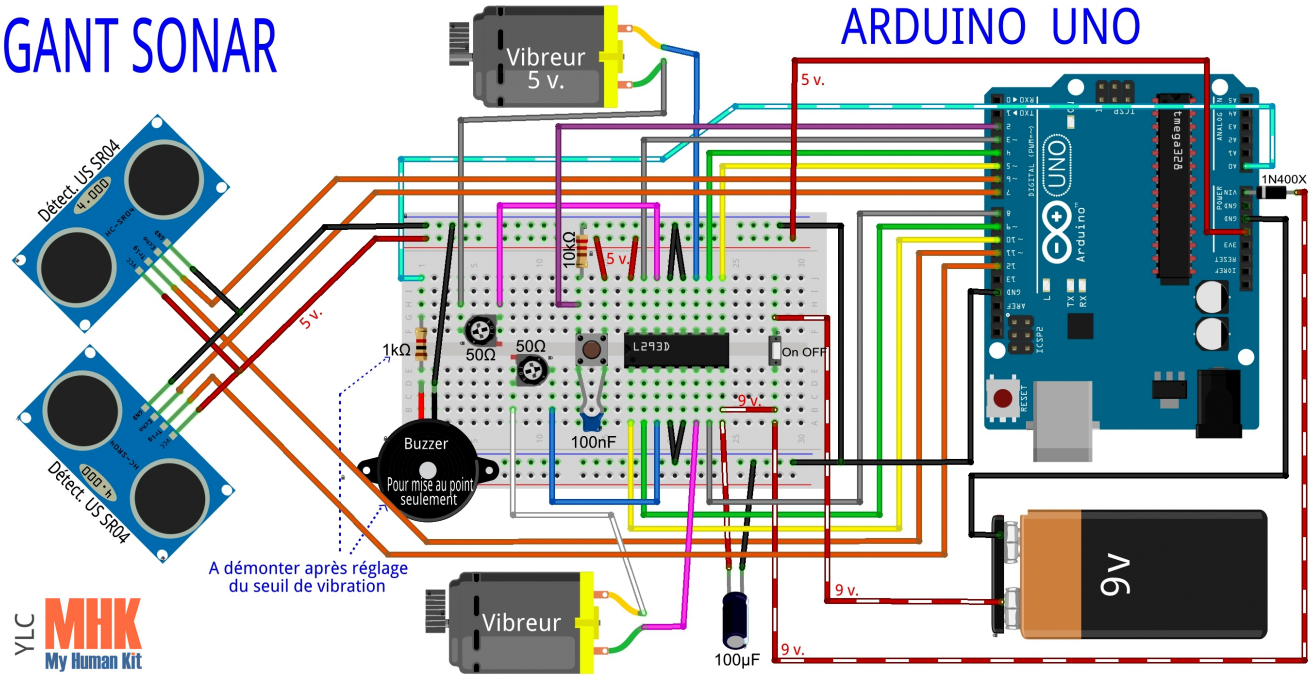
La distance maximale de détection avec le matériel décrit ici est de 2,50 m dans la pratique.
Comme la force des vibrations est inversement proportionnelle à la distance (plus l'obstacle est près et plus les vibrations sont fortes), il est parfois nécéssaire d'avoir une perceptions plus précise dans l'exploration d'un espace plus restreint.
On peut donc ajuster la distance maximale (correspondant donc au minimum de vibrations) à l'aide d'un bouton permettant de choisir entre une détection dans une plage courte jusqu'à 70 cm ou dans une plage longue jusqu'à 250 cm.

La distance à laquelle se trouve un obstacle est traduite par une vibration du côté de l'obstacle inversement proportionnelle à sa proximité. Plus l'objet est proche plus la vibration sera forte.
Contrairement à la version 1, la variation de distance d'un obstacle ne se traduit pas par une variation continue des vibrations mais par des variations par paliers. Il y a 4 paliers de vibrations qui permettent une meilleure appréhension de la variation de proximité des obstacles.
Ces paliers sont proportionnels à la plage de détection utilisée.
Pour une plage courte de 0 à 70 cm, les paliers de vibrations correspondent à des distances de 10, 25, 40, 70 cm.
Pour une plage longue de 0 à 250 cm les paliers de vibrations correspondent à des distances de 20, 80, 150, 250 cm.

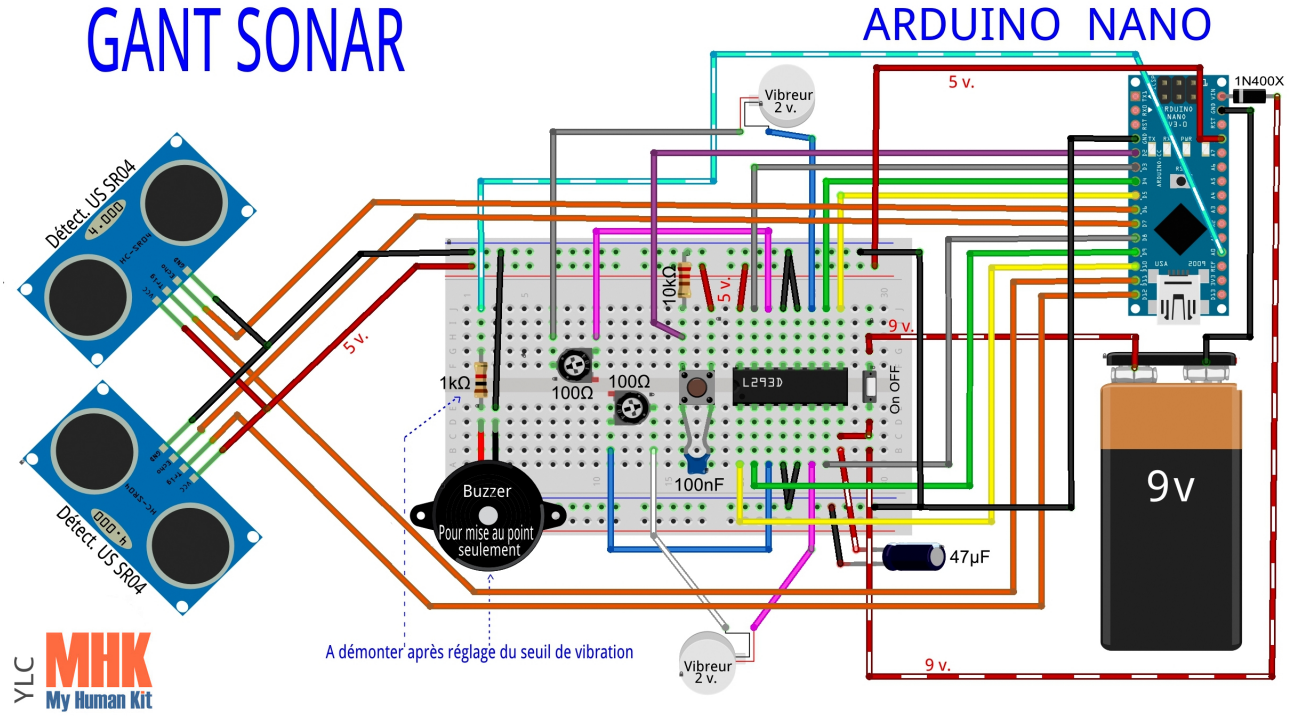
A noter que la gamme des forces de vibrations est définie par des paramètres qui eux-aussi pourraient être modifiés dans le programme selon la sensibilié de l'usager.

Shémas de montage sur breadboard V.1 et V2.1

Version basique avec un Arduino Uno et des vibreurs de console de jeux



Version de taille plus réduite avec un Arduino Nano et des vibreurs miniatures



Liste des composants V.1 et V2.1

(prix approximatif sur internet)

1 carte Arduino **UNO**		8 €
2 détecteurs ultrason HC-SR04	2 x 3 €	6 €
1 pont en H contrôle moteur DC L293D		3 €
1 bouton poussoir (OFF – ON momentané)		2 €
1 résistance (10 kΩ)		0,20 €
2 potentiomètres ajustable (50Ω 0,5 w)	2 x 1 €	2 €
1 condensateur céramique 100 ηF		0,20 €
1 condensateur chimique polarisé 100 µF		1,50 €
2 vibreurs de console de jeu	2 x 4 €	8 €
1 connecteur pile de 9v		1,40 €
1 interrupteur (on off)		2 €
1 pile 9v		4,50 €
1 diode 1N400x (protection polarité de la carte)		0,20 €
1 shield protypage UNO		9 €

Total : 48,00 € + fils + gant + support

ou

1 carte Arduino **NANO**		6 €
2 détecteurs ultrason HC-SR04	2 x 3 €	6 €
1 pont en H contrôle moteur DC L293D		3 €
1 bouton poussoir (OFF – ON momentané)		2 €
1 résistance (10 kΩ)		0,20 €
2 potentiomètres ajustable (100Ω 0,5 w)	2 x 1 €	2 €
1 condensateur céramique 100 ηF		0,20 €
1 condensateur chimique polarisé 47 µF		1,50 €
2 vibreurs miniatures (VM1201 Gotronic)	2 x 2,5 €	5 €
1 connecteur pile de 9v		1,40 €
1 interrupteur (on off)		2 €
1 pile 9v		4,50 €
1 diode 1N400x (protection polarité de la carte)		0,20 €
1 plaque veroboard pastilles		3 €

Total : 37,00 € + fils + gant + support

Comme on peut le voir ci-dessus, le coût est un peu moins élevé en utilisant une carte Arduino Nano plutôt qu'une Arduino Uno. Ceci est dû au coût de la carte, des moteurs et du shield (sauf récup et fabrication personnelle). Comme l'implantation du système sur le dos de la mains requiert plutôt des éléments de petite taille, on privilégiera ici un montage avec l'Arduino Nano bien que celui avec une Arduino Uno est tout aussi réalisable.

Notes de développement V2.1

Le programme a été entièrement réécrit par rapport à la V.1 afin de le rendre plus efficace et plus simple aussi.

Il a été développé sous IDE Arduino 1.8.1 (dernière version réputée stable).
Il utilise la bibliothèque newping qui permet un pilotage optimisé des capteurs à ultrasons et un meilleur temps de réponse.

Pour les vibreurs, on privilégiera des vibreurs de faible consommation et encapsulés dans un boitier pour un montage plus facile à réaliser. (*CF : vibreur VM1201 chez Gotronic*)

Les vibrations des deux vibreurs ne sont pas synchronisés et ils se déclenchent ou s'arrêtent indépendamment l'un de l'autre.

Une capacité de découplage de 100 µF permet d'absorber les montées en tension au démarrage des vibreurs et d'économiser ainsi la batterie.

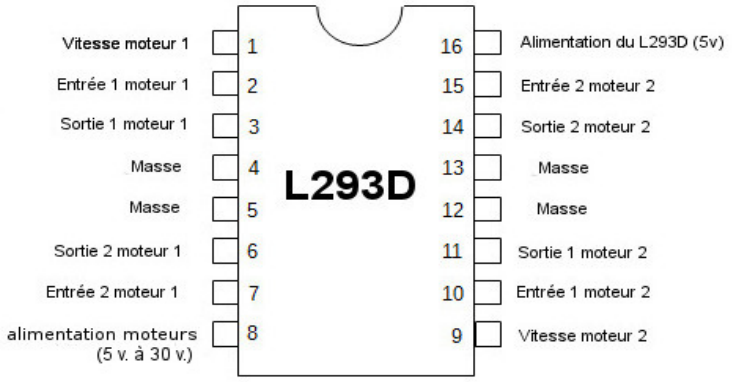
La distance maximale théorique est de 4 mètres pour les transducteurs à ultrasons SR04. Dans la pratique, on atteint tout juste 2,5 m. C'est pour cela qu'on a fixé la distance de détection maxi à 250 cm dans ce programme.
Il serait possible d'utiliser des transducteurs à ultrasons de portée supérieure (jusqu'à 6m) mais de coût nettement plus élevé aussi ou alors d'utiliser des mini-capteurs lidar d'une portée jusqu'à 12 m. mais nettement plus onéreux.

Le calcul de la distance réelle des objets n'a que peu d'importance puisque c'est leur distance relative par rapport aux mouvements de la main de l'utilisateur qui sont traduites en vibrations plus ou moins fortes. Le calcul des distances est donc fait sur la base d'une vitesse théorique du son de 340 m/sec.

Le sens de rotation de vibreurs n'a aucune importance et la perception reste la même quel que soit ce sens.

Pendant la phase de mise au point , afin de régler au mieux les paramètres de force des vibrations, ceux-ci étant étroitement liés à la conception des vibreurs, on peut utiliser provisoirement un buzzer couplé au sonar 1 pendant la phase de montage sur breadboard. Ce buzzer réagit par un son proportionnel à la distance et permet ainsi d'ajuster avec précision la valeur mini de vibration pour que le vibreur réagisse à la détection des objets les plus éloignés détectables. Il est connecté sur la pin D14 (≡ pin A0).
Le buzzer et sa résistance sont ensuite démontés après réglage et la constante correspondante est alors désactivée dans le programme.

ANNEXE 1 : Brochage du composant de commande des vibreur L293D

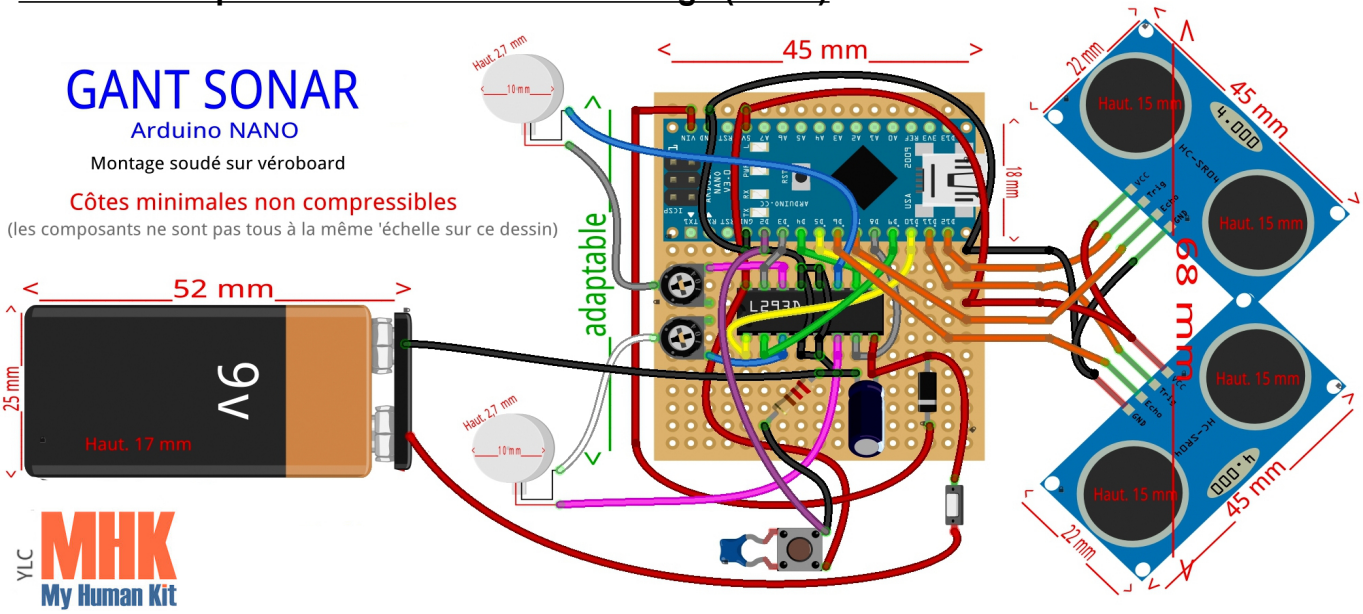


Commandes moteurs

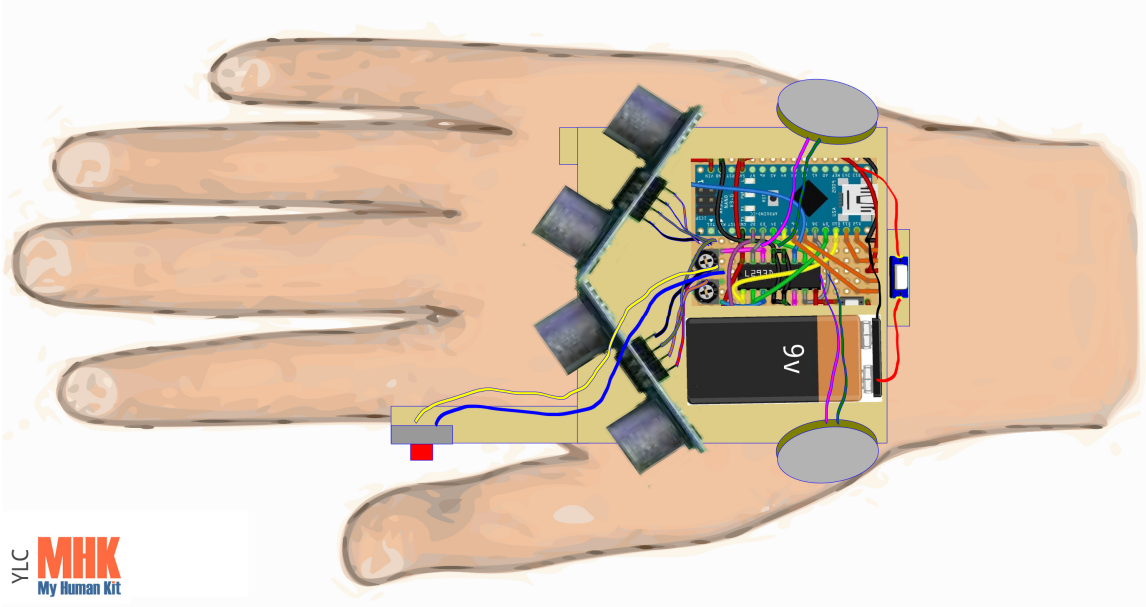
Vitesse	Entrée 1	Entrée 2	Résultat
> 0 V.	LOW	HIGH	Rotation horaire
> 0 V.	HIGH	LOW	Rotation anti-horaire
> 0 V.	LOW	LOW	Moteur arrêté
> 0 V.	HIGH	HIGH	Moteur arrêté
0 V.			Moteur arrêté

NB : les broches « vitesse » sont pilotées en PWM.

ANNEXE 2 : plan des contraintes de montage (cotes)

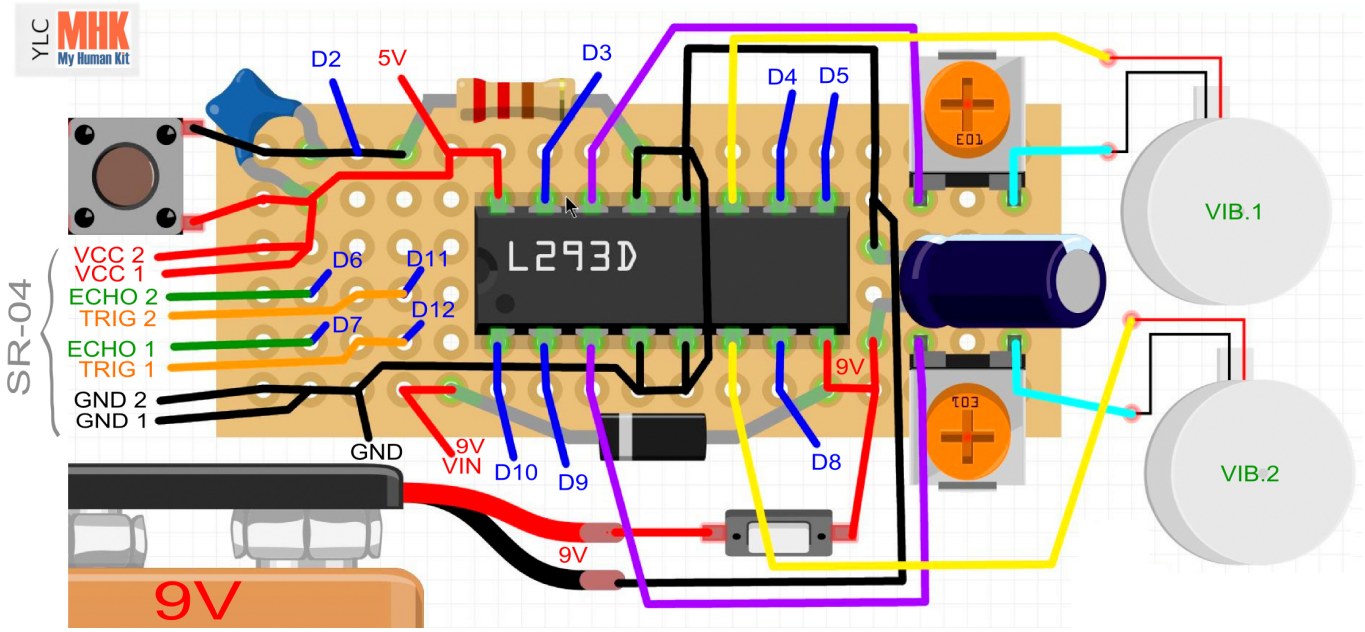


ANNEXE 3 : Positionnement des éléments sur la main

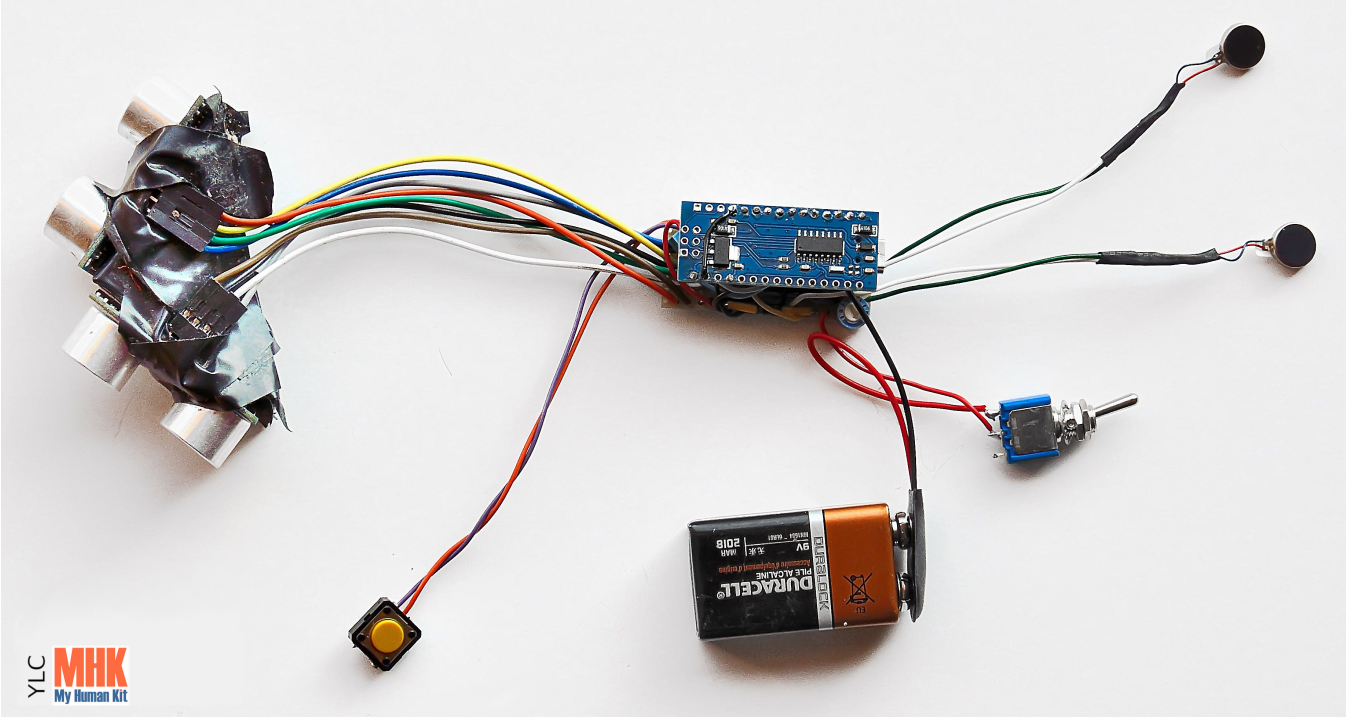


L'implantation représentée ci-dessus est celle d'un droitier.
Le support du bouton doit pouvoir se positionner de l'autre côté pour un gaucher.

ANNEXE 4 : shéma des connections du prototype NANO V.1 et V2.1



ANNEXE 5 : prototype NANO sans support V.1 et V2.1



NB : dans ce montage «dessus-dessous» avec véroboard sur une carte NANO, l'ensemble du montage (arduino + composants) a la même longueur et la même épaisseur que la pile de 9v. mais est moins large de 5 mm. Soit 52 mm de long, 17 mm d'épaisseur et 20 mm de large. (prévoir cependant des marges d'au moins 1mm en plus).

ANNEXE 6 : Programme Arduino V2.1

```
/*********************************************************************
GantSonar v2_1        Version simplifiée ... mais plus efficace  :-))
Dispositif haptique portatif pour la perception de l'environnement par des mal-voyants
Matériel utilisé : Arduino NANO + (2 x HC-SR04) + LD293D + (2 x vibreurs)
Principe : l'écho des objets est rendu par des vibrations plus ou moins fortes pour indiquer la proximlité d'un objet
Choix de 2 plages de détection : courte de 0 à 70 cm et longue de 0 à 250 cm (bascule du choix par bouton)
Dans chaque plage, les variations des vibrations se font par 4 paliers pour une meilleure perception de l'accroissement de proximité.
Développement : Yves Le Chevalier pour My Human Kit - révision 08/2016
//-----------------------------------------------------------------
NB : UTILISATION DE LA LIBRARY NEWPING (meilleur temps de réponse des sonars)
*********************************************************************/
#define TRACES 0   // mise au point : 1 pour activer trace sur port série, 0 pour désactiver
//-----------------------------------------------------------------
#include <NewPing.h>       // testé avec la Lib Newping V 1.8

// VARIABLES DISTANCES
#define plagedetect 2   // pin D2 : bouton bascule de la plage de détection courte-longue (interruption 0 sur pin D2)
unsigned long timpre, timbou = 0; // variable de temps pour éviter le rebond sur appui bouton
int plage = 0;         //  indicateur plage de mesure des distances choisie (0 = courtes par défaut)
int paldis[2][4] = { {10, 25, 45, 70}, {20, 80, 150, 250} };   //  4 paliers des distances courtes et longues (cm)
int distmax = paldis[plage][3];     // variable de calcul distance maxi selon la plage

// VARIABLES FORCES DE VIBRATIONS
int palvib[4] = {255, 140, 80, 50};   // forces de vibrations correspondant aux 4 paliers distances de chaque plage

// SONAR GAUCHE
#define ptrigg 11   //  pin D11 : trigger d'activation du sonar gauche
#define pechog 6    //  pin D6 : renvoi une durée d'écho proportionnelle à la distance
int distg;          // distance mesurée par sonar gauche
NewPing sonarg(ptrigg, pechog, distmax);

// SONAR DROIT
#define ptrigd 12   //  pin D12 : trigger d'activation dur sonar droit
#define pechod 7    //  pin D7 : renvoi une durée d'écho proportionnelle à la distance
int distd;          // distance mesurée par sonar droit
NewPing sonard(ptrigd, pechod, distmax);

// VIBREUR GAUCHE
#define penabg 5    // pin D5 : commande vitesse vibreur gauche (pin 1 du L293D)
#define pinpug1 4   // pin D4 : commande 1 sens de rotation (pin 2 du L293D)
#define pinpug2 3   // pin D3 : commande 2 sens de rotation (pin 7 du L293D)
int valvibg;        // force de vibration à appliquer

// VIBREUR DROIT
#define penabd 10   // pin D10 : commande vitesse vibreur droit (pin 9 du L293D)
#define pinpud1 9   // pin D9 : commande 1 sens de rotation (pin 10 du L293D)
#define pinpud2 8   // pin D8 : commande 2 sens de rotation (pin 15 du L293D)
int valvibd;        // force de vibration à appliquer

// ***************************** SETUP ******************
void setup() {
#if TRACES        // Trace pour la mise au point du programme sur port série
  Serial.begin(9600);
  Serial.println("Gant sonar v2_0");
#endif
  pinMode(ptrigg, OUTPUT);     // initialisation du sens des pins de la carte
  pinMode(ptrigd, OUTPUT);
  pinMode(pechog, INPUT);
```

```
  pinMode(pechod, INPUT);
  pinMode(penabg, OUTPUT);
  pinMode(pinpug1, OUTPUT);
  pinMode(pinpug2, OUTPUT);
  pinMode(penabd, OUTPUT);
  pinMode(pinpud1, OUTPUT);
  pinMode(pinpud2, OUTPUT);
  digitalWrite(pinpug1, LOW);   // pin 1 low + pin 2 high ...
  digitalWrite(pinpug2, HIGH);  //        ... vibreur gauche tourne dans sens horaire
  analogWrite(penabg, 0);       // vibreur gauche arrêté au départ
  digitalWrite(pinpud1, HIGH);  // pin 1 high + pin 2 high ...
  digitalWrite(pinpud2, HIGH);   //        ... vibreur droit tourne dans sens horaire
  analogWrite(penabd, 0);       // vibreur droit arrêté au départ
  attachInterrupt(0, memodist, RISING );   // bouton bascule plage détection sur broche 2 (interruption #0)
}
// ***************************** MAIN LOOP ******************
void loop() {
  distg = sonarg.convert_cm(sonarg.ping_median(3, distmax));  // mesure distance sur sonar gauche
   if (distg > 0 && distg < distmax)   valvibg = calcforce(distg);     // calcul force vibration 1 pour la distance 1
mesurée...
  else valvibg = 0;                                          // ... si mesure dans la plage choisie
  analogWrite(penabg, valvibg);        // activation ou arrêt du vibreur gauche

  distd = sonard.convert_cm(sonard.ping_median(3, distmax));  // mesure distance sur sonar droit
   if (distd > 0 && distd < distmax)   valvibd = calcforce(distd);     // calcul force vibration 2 pour la distance 2
mesurée...
  else valvibd = 0;                                          // ... si mesure dans la plage choisie
  analogWrite(penabd, valvibd);       // activation ou arrêt du vibreur droit

  #if TRACES
    Serial.print(" dist gauche : ");
    Serial.print(distg);
    Serial.print(" cm     dist droite : ");
    Serial.print(distd);
    Serial.print(" cm   distmax : ");
    Serial.print(distmax);
    Serial.print("  Force gauche : ");
    Serial.print(valvibg);
    Serial.print("  Force droite : ");
    Serial.println(valvibd);
  #endif
}
//---------------------------------------------------------
int calcforce(int dist) {       // calcul force de vibration en fonction inverse de la distance...
  int forvib = 0;               // ... selon le palier de mesure avec la force correspondante à ce palier
  if (dist <= paldis[plage][0]) forvib = palvib[0];
  else if (dist <= paldis[plage][1]) forvib = palvib[1];
       else if (dist <= paldis[plage][2]) forvib = palvib[2];
            else forvib = palvib[3];
  return forvib;
}
//---------------------------------------------------------
void memodist() {        // routine d'interruption pour choisir la plage mesurée
  timbou = millis();               // memorisation temps actuel
  if (timbou - timpre > 1000) {      // temporisation 1 sec anti-rebond du bouton
    if (plage == 0)  plage = 1;      // mesure plage distances longues
    else plage = 0;                  // mesure plage distances courtes
  }
  timpre = timbou;                   // mémorisation de l'heure d'appui sur le bouton
  distmax = paldis[plage][3];        // définir distance maxi selon plage choisie
}
// ********************************** FIN ***************************************
[/code]
```