Réflexions et prise de décisions

A l’issue de ce projet nous souhaitons obtenir un robot allant de A à B en suivant une trajectoire tout en évitant des obstacles et sans jamais connaître l’espace de navigation au complet.

Le robot ne va donc pas se contenter de changer de direction à la détection d’un obstacle mais il va le contourner afin de revenir sur sa trajectoire.

Une image contenant texte, carte

Description générée avec un niveau de confiance très élevé

# Les capteurs

## Nombre de capteurs

Pour ce projet nous savons d’avance que nous serons face à un problème de mémoire de notre carte Arduino pour l’implémentation de l’algorithme A\*. De ce fait, nous choisissons de n’utiliser pour l’instant que 3 capteurs afin de ne pas avoir de surprise lors de l’implémentation. Si cela est possible nous pourrons créer un deuxième prototype avec plus de capteurs.

Comment permettre au robot de longer un obstacle et de détecter ceux qui sont en biais et cela avec seulement 3 capteurs ?

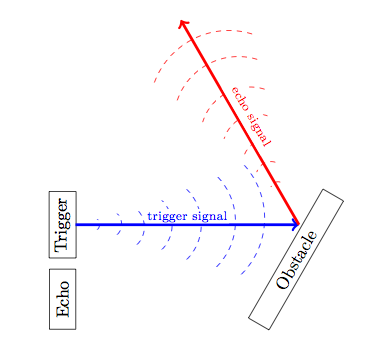
## Position des capteurs

Une image contenant capture d’écran

Description générée avec un niveau de confiance élevé

La principale réflexion ici est de connaître exactement l’angle de détection des capteurs afin de connaître l’angle A, d’estimer la distance maximale entre l’obstacle et le robot pour être détecté et enfin de prévoir une distance minimale entre l’obstacle et le robot pour que celui-ci puisse le dévier sans avoir à reculer et/ou entrer en collision.

Dans cette configuration il n’y a pas d’interférences possible entre les capteurs ultrasons, mais la position des obstacles est à prendre en compte. En effet, si un obstacle est à de plus de 30° du capteur il émettra un signal mais ne recevra pas de réponse. Comme expliqué ci-dessous extrait de <https://macduino.blogspot.com/2013/11/HC-SR04-part1.html> .

[](https://1.bp.blogspot.com/-XBTy4BdExjA/UoHreFiHbTI/AAAAAAAAAHo/hlZm_cywuBQ/s1600/SensorSignalDeflection.png)Le son est une onde longitudinale (c'est-à-dire qu'il progresse le long d'une ligne horizontale). Par conséquent, lorsque l'obstacle n'est pas parfaitement devant le module, les sons sont déviés et le signal d'écho risque de ne pas atteindre le capteur ni de l'atteindre très atténué et donc de ne pas être détecté (voir la figure ci-dessous).

Lorsque l'obstacle est à un angle supérieur à 30 ° par rapport à la direction de propagation du signal de déclenchement, le capteur donne des résultats erratiques.

Ainsi selon le constructeur et en se référant aux tests d’autres chercheurs nous allons nous référer à ces chiffres pour notre premier prototype.

## Le capteur HC-SR04

**Caractéristiques :**

Dimensions : 45 mm x 20 mm x 15 mm  
Plage de mesure : 2 cm à 400 cm  
Résolution de la mesure : 0.3 cm  
Largeur d'impulsion sur l'entrée de déclenchement : 10 μs (Trigger Input Pulse width)

Angle de mesure efficace : 15 ° (de chaque côté) donc 30°   
 entre chaque capteur

**Broches de connexion :**

Vcc= Alimentation +5 V DC  
Trig= Entrée de déclenchement de la mesure (Trigger input)  
Echo= Sortie de mesure donnée en écho (Echo output)  
GND= Masse de l'alimentation

En utilisant ces données nous rejoignons la réflexion retrouvée ici : <https://macduino.blogspot.com/2013/11/hc-sr04-using-multiple-ultrasonic.html> . Ainsi nous pouvons nous baser sur les calculs de Mr Formenti pour définir la position exacte de nos capteurs suivant le schéma suivant :

Une image contenant texte

Description générée avec un niveau de confiance très élevé

# Le premier prototype

La position des capteurs sur notre prototype

Les pièces du robot nous étant imposé pour une première version, nous avons dû nous adapter pour la position des capteurs. Au lieu de les positionner selon la réflexion vue au paragraphe précédent, nous les avons placés avec un angle de 50 à 60 degrés d’écart chacun.

La distance entre le capteur et l’obstacle

**Distance = ([Durée du niveau haut] \* [vitesse du son :340m/s]) / 2** représente la distance entre le capteur et l’obstacle où **Durée du niveau haut** représente la durée de l’impulsion durant au moins 10μs pour que le module démarre sa lecture.

On peut simplifier cette formule grâce aux informations du constructeur :

**Distance (cm) = impulsion (µs) / 58**

Un programme de calcul de distance est disponible sur le github  ***arduino/calculDistance.ino***

Après la première réunion de suivi de projet, on nous a conseillé de se focaliser sur un robot qui revient à son point de départ plutôt qu’un robot qui va d’un point A vers un point B.