Rapport de projet

Système de recul automobile

Table des matières

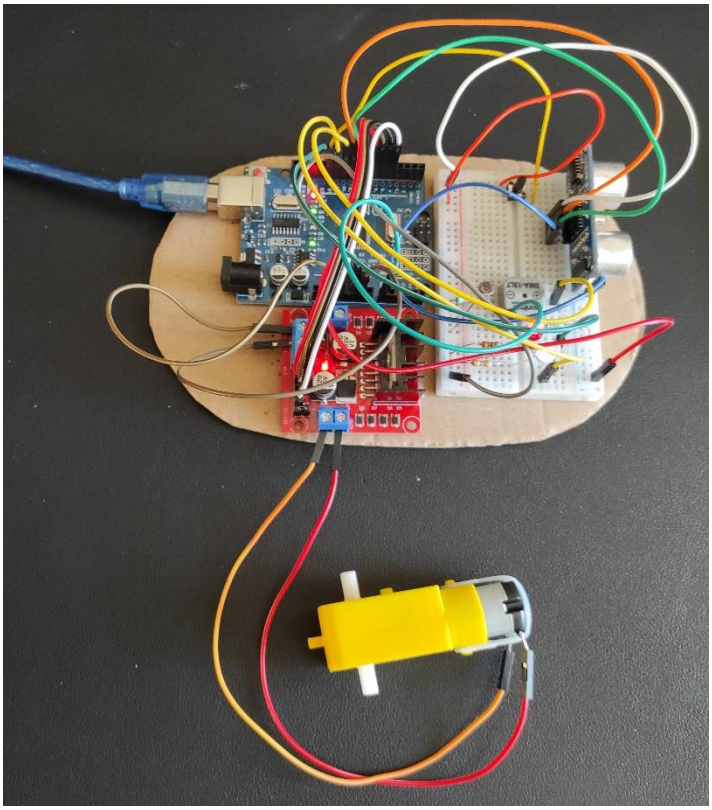


Photo de notre projet

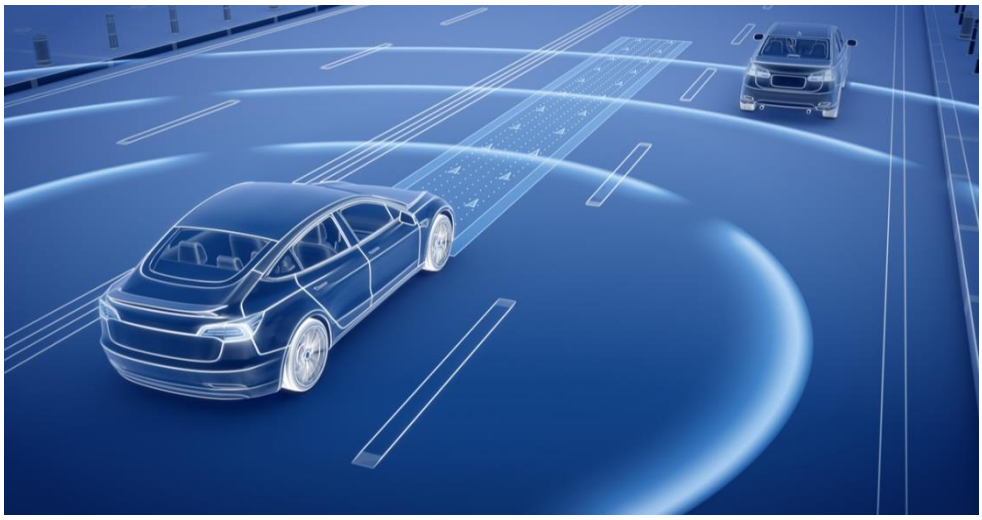
1. Présentation du projet :

Nom : Système de recul automobile et phare intelligent

Participant : Noé Game et Ugo Bernard

Description du projet et cahier des charges :

1) Le système doit être capable détecter la distance d'un obstacle et de faire allumer une led (dite voyant) ainsi que le buzzer à une fréquence d'autant plus rapide que l'obstacle est proche. De cette manière une personne malentendante bénéficie aussi de ce système automobile.



Radar de voiture

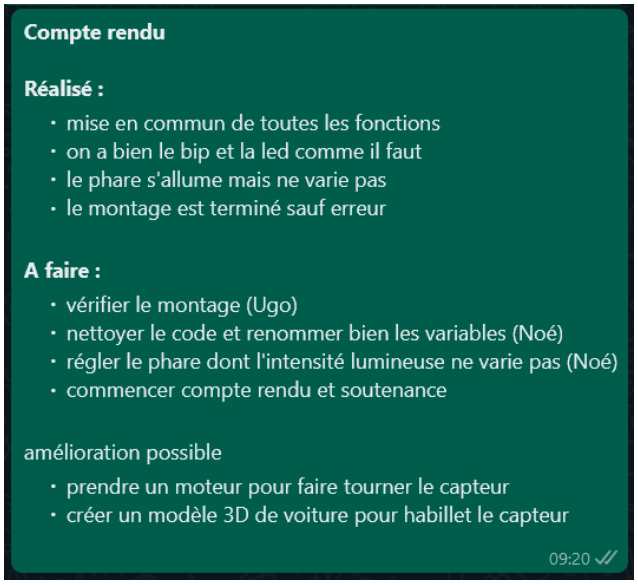
2) Le système doit être capable de détecter l'intensité lumineuse et allumer une led (dite phare) plus ou moins fortement selon la quantité lumière présente, afin de recréer un système de phare intelligent et économe en énergie

Phare intelligent

2. Les problèmes rencontrés et les solutions élaborées

2.A Travailler en équipe

Compte rendu :

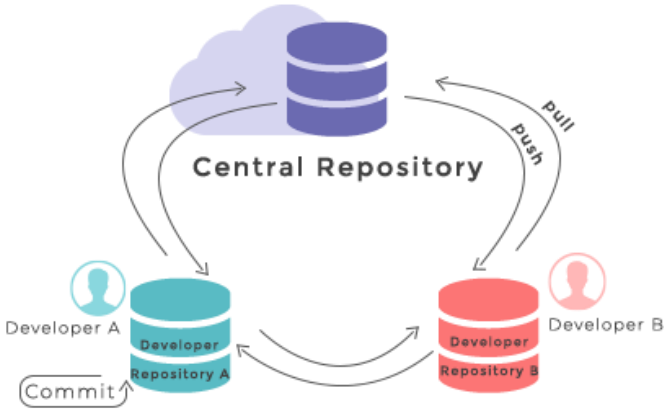


Exemple de compte rendu envoyé sur WhatsApp

Nous avons mis en place des solutions apprises avec Phillipe Rouffie afin de mieux nous organiser au sein de notre équipe. Notamment avec la mise en place d'un **compte rendu** réalisé pendant les 5 dernière minutes de chaque séance afin de se rappeler à la séance d'après ce que nous avions réalisé la dernière fois ainsi ce qu'il nous reste à faire.

Utilisation d'un git :

Qu'est-ce qu'un dépôt Git ? Un dépôt (repository en anglais) Git est un espace de stockage centralisé ou décentralisé qui contient l'historique complet des modifications apportées à un projet. Il conserve les différentes versions du code source, permet le suivi des changements, et facilite la collaboration entre les membres d'une équipe de développement. Un dépôt peut être local (sur l'ordinateur de l'utilisateur) ou distant (sur un serveur), et il est essentiel pour gérer efficacement le développement de logiciels en utilisant Git.



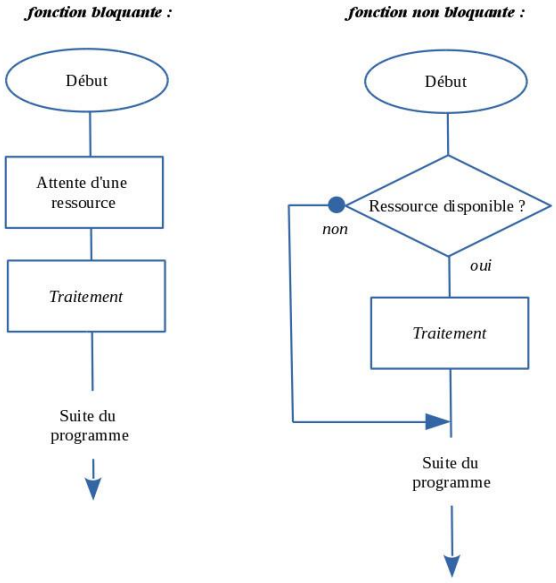
Fonctionnement d'un git

Nous avons donc mis en place cette solution afin de pouvoir écrire du code tous les deux ensembles sur les mêmes fichiers, ce qui nous a était très pratique pour pouvoir écrire du code en même temps, ou à des moments différents, au même endroit ou séparément. Le fichier contenant notre code à jour n'est plus bloqué sur un ordinateur, mais devient accessible sur tous. Néanmoins cela n'a pas été facile au début car nous ne savions pas utiliser ni Git ni GitHub, le serveur qui héberge notre dépôt, malgré une formation au club robot.

Notre dépôt : https://github.com/eseonoegame/project_arduino_car_reversing_system

2.B Programmation non bloquante

Une autre problématique que nous avons rencontrée a été celle de la programmation non bloquante, problématique déjà rencontré lors de l'exercice du feu tricolore.



Programmation bloquante et non bloquante

Qu'est-ce que la programmation non bloquante ? Pour comprendre ce qu'est la programmation non bloquante, imaginez que vous deviez allumer et éteindre une led tous les 2 secondes. Une boucle et un sleep() réglerais rapidement et facilement ce simple problème. Mais maintenant si je vous demandais de faire la même chose avec une dizaine de led et des intervalles de temps aléatoire, ce ne serait pas la même histoire.

Effectivement la fonction sleep() n'est alors plus utilisable, car si l'on utilise la fonction sleep(), le programme ne fera strictement rien durant cet intervalle de temps, ce qui est embêtant si une led devait être allumé à ce moment-là. La fonction sleep est alors considéré comme bloquante.

La programmation non bloquante consiste donc à ne pas utiliser de fonction dite bloquante, comme la fonction sleep().

Allumer plusieurs led à la fois

Mais en quoi cela concerne notre projet me direz-vous ? Eh bien notre programme doit principalement tester en boucle si la distance de l'obstacle implique de changer la fréquence d'allumage de notre buzzer, ainsi que l'intensité lumineuse de la led en fonction de l'intensité lumineuse extérieur. Mais il ne faudrait pas qu'au prétexte que le programme regarde s'il fait beau dehors, le programme arrête d'actualiser la fréquence d'allumage du buzzer. Cela pourrait être dangereux, l'utilisateur pourrait, n'entendant pas de son, reculer rapidement vers un obstacle qui n'aurait pas été signalé par la voiture et alors créer un accident.

Il faut donc que notre programme vérifie nos deux paramètres en permanence et a une fréquence assez élevée.

```
void loop() // appellé en boucle par le processeur.
{
  distance = getDistance(); // Recupere la distance mesure par le capteur.
  testDuree(distance);  // Test si la distance implique de changer la frequence d'activation du buzzer.
  testBuzzer(distance); // Test si il est temps de changer l'etat du buzzer (on/off). Buzz avec une fréquence qui dépend de la periode allume.
  testSun();            // Test si la lumière extérieur implique de changer l'intensité lumineuse du phare.
  debug();              // Affiche etats des variables afin de debuguer le code.
}
```

Notre code, appelle en boucle par le processeur plusieurs fois par seconde

2.C Utiliser le capteur sans librairie

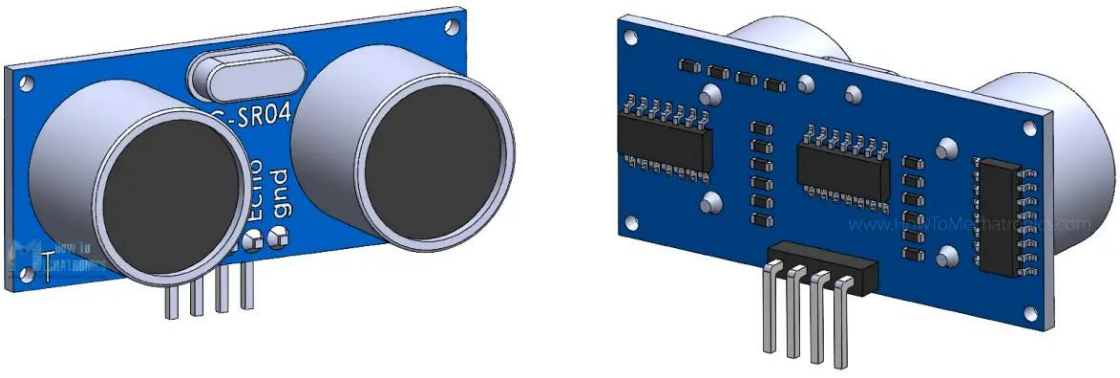
La deuxième difficulté rencontrée et pas la moindre était de faire fonctionner le capteur. La première chose que nous avons faite a été de copier-coller un code tout fait sur internet qui a fonctionné directement après avoir installé la librairie qui allait bien.

Mais nous n'étions pas satisfaits de cette manière de fonctionner et nous avons voulu comprendre comment faire fonctionner le capteur avec les fonctions natives Arduino. Après un rapide rappel des cours de physique de terminal sur la vitesse de déplacement d'une onde et quelques explications trouvées sur internet, nous avons réussi à réécrire par nous même une fonction qui récupère la distance d'un obstacle à l'aide du capteur.

Notre programme fonctionne de manière très simple, tout d'abord nous envoyons une onde pendant 10 microsecondes, en alimentant une broche du capteur, puis nous écoutons le temps que le met l'onde à revenir au récepteur de notre capteur. Ce temps multiplié par la vitesse de propagation de l'onde nous renvoi la distance. Mission accomplie !

```
int getDistance(void)  // retourne la distance de l'obstacle détecté par le capteur.
{
  int d;
  // Émission d'un signal de durée 10 microsecondes
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Écoute de l'écho
  dureeEcho = pulseIn(echoPin, HIGH);
  // Calcul de la distance
  d = dureeEcho * 0.034 / 2;  //distance = 1/2 vitesse du son fois la durée
  return d;
}
```

Notre fonction pour interagir avec le capteur



Le capteur Arduino

2.D Varier l'intensité lumineuse d'une led

Nous avions comme ambition de faire varier l'intensité lumineuse de deux leds. La première, dite « voyant » devait s'allumer d'autant plus intensément qu'un obstacle est proche. Cela permet à une personne malentendante de bénéficier aussi de notre système de recul automobile. Mais aussi une seconde led dite « phare » afin d'éclairer plus ou moins fort selon la lumière extérieure.

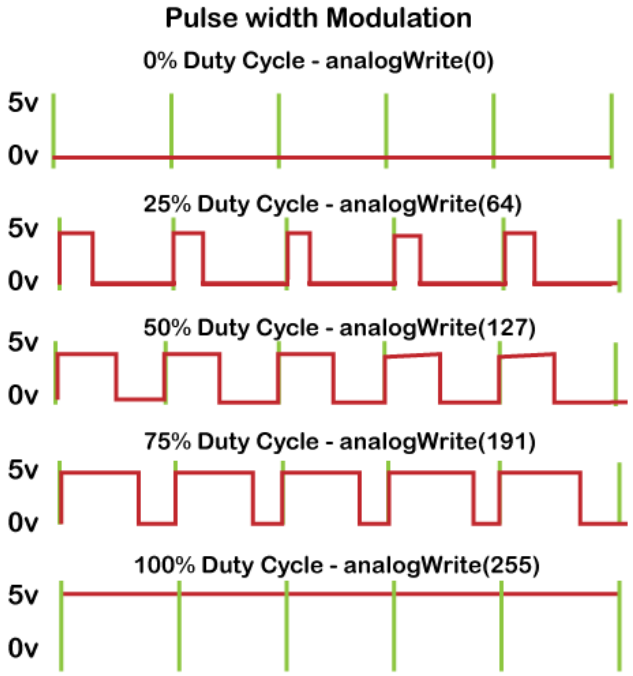
Nous avons donc créé une fonction pour reproduire le principe de Pulse With Modulation (PWM), c'est-à-dire d'allumer et éteindre la led à une fréquence d'autant plus élevée que nous souhaitions une intensité lumineuse élevé.

Notre fonction fonctionnait bien de manière isolée, mais une fois intégré dans notre code elle fonctionnait moins bien, car elle était appelée des centaines de fois par seconde comme les autres fonctions du programme, ce qui créait une petite latence compliquée à gérer

Nous avons finalement utilisé les ports analogiques de la carte Arduino, qui fonctionne aussi en réalité avec la méthode PWM, mais qui est prise en charge par la carte et donc plus simple à utiliser.

La variation d'intensité lumineuse n'étant pas très visible sur les leds rouges, nous avons préférés les faire clignoter au rythme du buzzer.

Pour savoir à quelle intensité lumineuse allumé le phare, nous mesurons juste la tension en sortie de notre capteur de luminosité, qui est simplement une résistance qui varie en fonction de la quantité de lumière, puis nous appliquons une simple règle de 3.



Fonctionnement du PWM

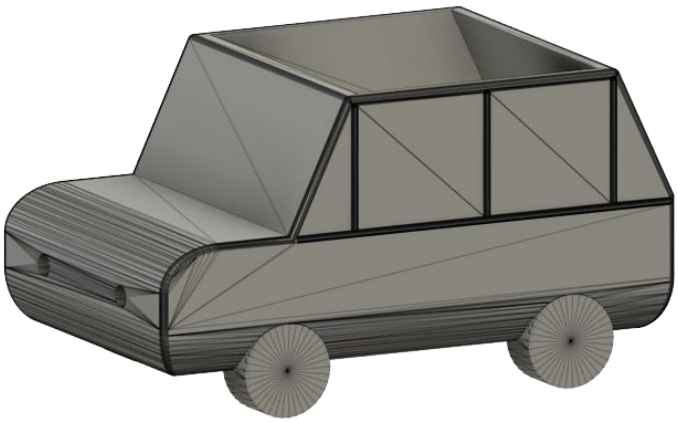
2.E Impressionnante impression 3D

Nous avons réalisé que notre projet était sympa mais qu'il le serait encore plus dans une voiture. Nous avons alors pensé à fusionner notre projet avec celui de Nicolas et Pénélope, qui se sont lancé dans la construction d'une voiture, mais devant les difficultés d'organisation et de conception de nos projets, nous n'avons finalement pas aboutit à une collaboration.

Cependant aux formations proposées par le club robot, nous avions appris à utiliser un dépôt git, mais aussi un logiciel de modélisation 3D : SolidWorks. Nous nous sommes alors dit que notre projet serait plus joli dans une voiture imprimée en 3D. Ce serait une première pour nous et le club EseoLab propose d'imprimer les projets que les étudiants leurs envoi. Devant cet alignement d'étoiles nous n'avons pas su résister et avons sauter sur l'occasion. Nous cherchions un moyen d'améliorer notre projet électroniquement mais nous ne voyions pas d'amélioration possible à un système de recul automobile, un concept déjà bien établi.

La conception n'a pas été simple ; il a fallu apprendre à maitriser un logiciel professionnel, les règles de l'impression 3D. Nous en avons retenu une leçon : faire une esquisse papier avant de se lancer pour ne pas recommencer plusieurs fois inutilement !

Une fois notre projet finit, devant le temps d'impression requis, notre impression a été refusé, car l'objet que nous voulions imprimer ne présentait une vraie fonctionnalité pratique qui aurait justifié ce temps d'impression



Modèle 3D vue devant 1

Modèle 3D vue derrière 1

On observer sur deux emplacements à l'avant pour les leds dites « phare » ainsi que deux emplacements à l'arrière pour les leds dites « voyants » de notre système. Les deux emplacements plus larges devaient contenir le capteur.

2.F Plan B

En regardant notre projet sans sa belle enveloppe, il nous semblait tout de suite moins impressionnant, alors nous avons décidé d'améliorer notre circuit électronique. Nous avons alors ajouté un moteur pour symboliser la motorisation de notre véhicule. Le moteur s'arrête si la distance à un obstacle est trop faible, afin d'éviter toute collision. Un bouton est présent pour pourvoir changer de sens de rotation du moteur, afin de faire avancer ou reculer notre voiture.

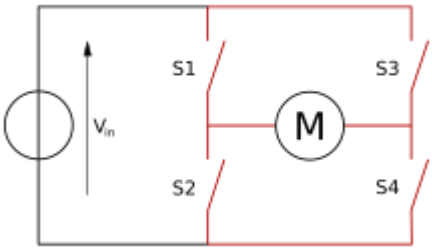
Nous avions construit notre code d'un telle manière que pour ajouter de nouvelles fonctionnalités, il suffit de les rajouter dans sous forme d'appel dans de fonction dans loop() sans que cela impact les autres fonctionnalités déjà mise en place. Voir la partie précédente sur la programmation non bloquante.

```
void loop() // appellé en boucle par le processeur.
{
  distance = getDistance(); // Recupere la distance mesure par le capteur.
  testDuree(distance);  // Test si la distance implique de changer la frequence d'activation du buzzer.
  testBuzzer(distance); // Test si il est temps de changer l'etat du buzzer (on/off). Buzz avec une fréquence qui dépend de la periode allume.
  testSun();            // Test si la lumière extérieur implique de changer l'intensité lumineuse du phare.
  testMoteur(distance); // Test si la distance implique de changer le sens de rotation sens du moteur.
  testBouton();         // Test si une pression sur le bouton implique de changer sens rotation du moteur.
  debug();              // Affiche etats des variables afin de debuguer le code.
}
```

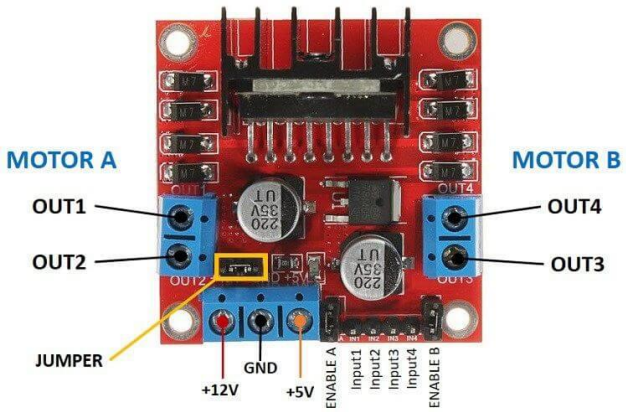
Ajout de la fonction testMoteur() et testBouton().

Nous avons rencontré une difficulté concernant le moteur : comment inverser le sens de rotation de celui-ci ? Le fonctionnement d'un moteur est pourtant simple : le 5V d'un coté et la masse de l'autre. Pour inverser le sens de rotation du moteur, il suffit d'inverser la masse et le 5V. Oui mais comment faire ? Peut-on dire à la carte de convertir une broche un coup en 5V et un coup en GND ? A priori non, du moins nous n'avons pas trouvé, mais nous avons trouvé une autre solution : le pont en H.

Le pont en H est un simple montage électronique qui permet d'inverser la masse et le GND à l'aide d'interrupteur. Pour ce faire nous avons utilisé un « pilote de moteur pas à pas L298N »

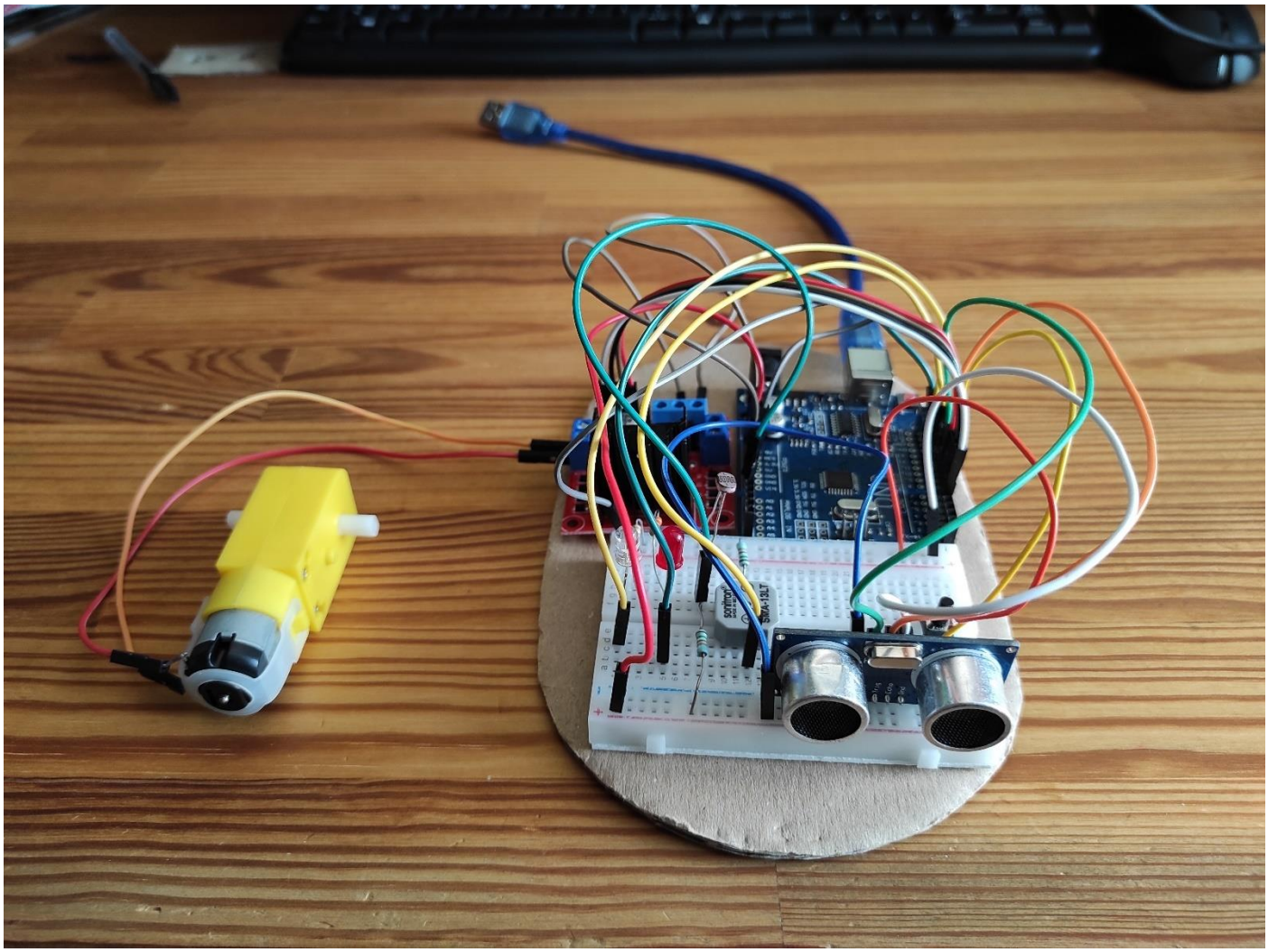


Pont en H, Wikipédia.



Pilote de moteur pas à pas L298N

3 Résultats



Le système est bien capable détecter la distance d'un obstacle et de faire allumer une led (dite voyant) ainsi que le buzzer à une fréquence d'autant plus rapide que l'obstacle est proche. De cette manière une personne malentendante bénéficie aussi de ce système automobile.

Le système est bien capable de détecter l'intensité lumineuse et allumer une led (dite phare) plus ou moins fortement selon la quantité lumière présente, afin de recréer un système de phare intelligent et économe en énergie

Le système arrête bien le moteur si la distance à un obstacle est trop faible, afin d'éviter toute collision. Un bouton est présent pour pourvoir changer de sens de rotation du moteur, afin de faire avancer ou reculer notre voiture.