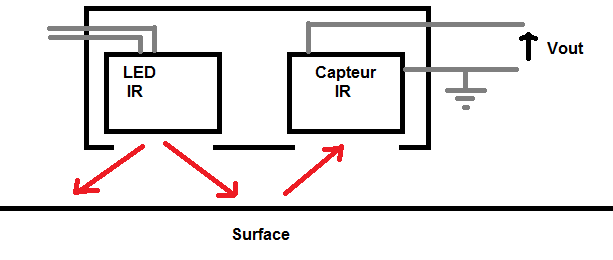
**DOC Capteur de distance**

**Capteur IR**

**Fonctionnement :**

Un capteur de distance est un dispositif, qui comme son nom l’indique, permet de mesurer la distance entre le capteur et un objet cible (qui peut être une personne, un animal, un véhicule, etc).



Ce schéma permet de mieux visualiser le fonctionnement du capteur.

Sur la **gauche** on voit que la **LED infrarouge** émet une lumière vers la surface réfléchissante.

Et la **photodiode** a **droite** reçoit plus ou moins la lumière réfléchit.

Selon la couleur de la surface ainsi que sa distance.

Utilisation

Ce capteur peut avoir différentes fonctions :

* Calculer une distance
* Détecter un changement de contraste
* Mesurer la réflectivité d’un milieu en fonction de la distance

**Compteur de vitesse**

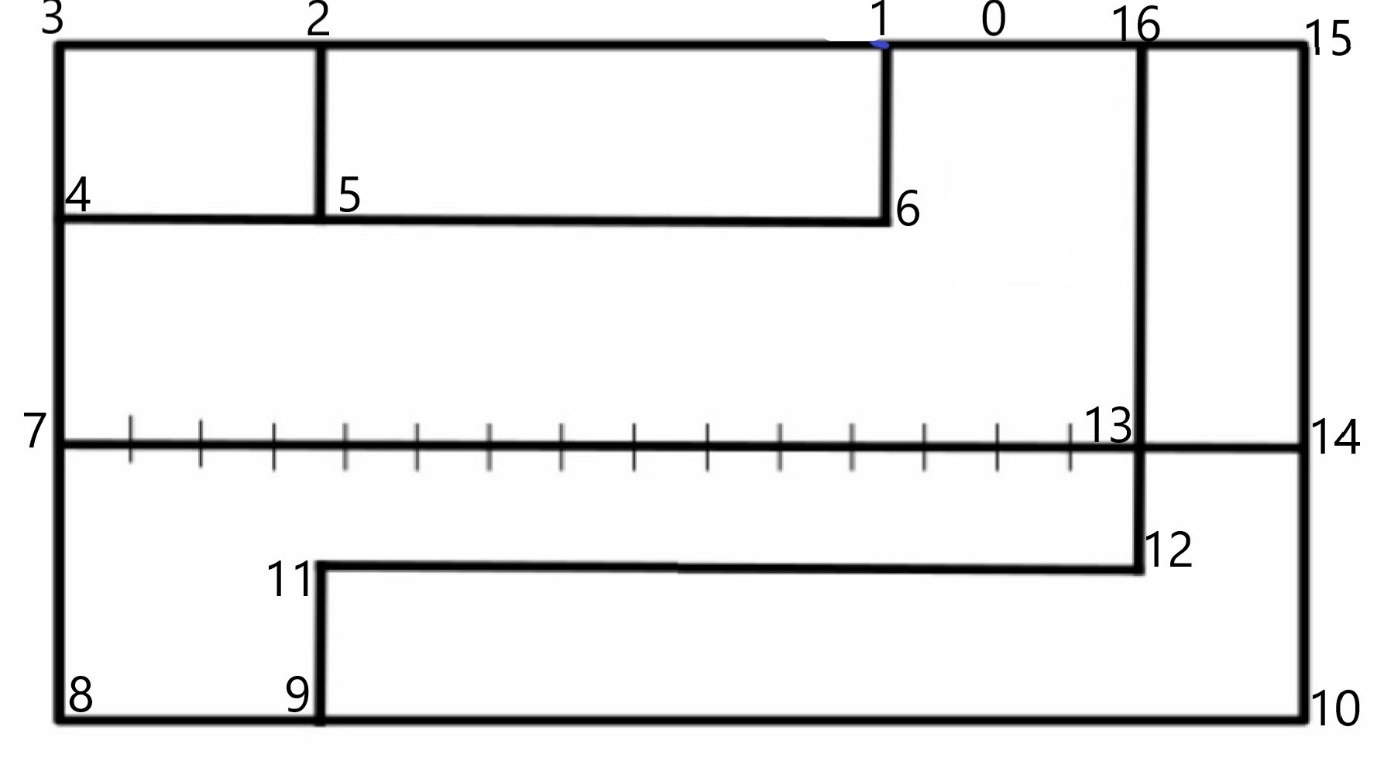
**Fonctionnement :**

A l’aide d’un disque de comptage on peut mesurer la vitesse d’un moteur. Peut aussi servir à détecter un portail qui se ferme, etc.

**Pathfinding**

Pathfinding est un terme anglais qui peut se traduire en « **Trouver son chemin** ».

Notre véhicule devra donc être capable à partir d’un point de départ A de se déplacer jusqu’au point B en utilisant le chemin le plus court tout en suivant la route.



**Notre map**

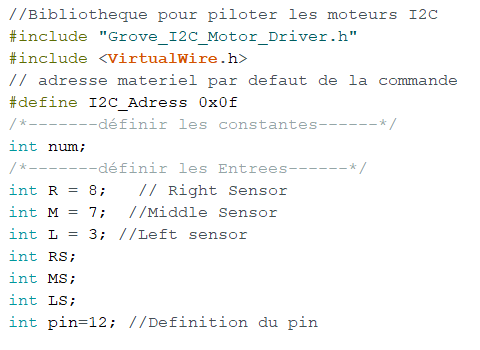
Comme le montre cette photo, les ***surfaces vertes représentent des places de parking***.

Notre but et donc de ***détecter ces places*** en détectant le ***contraste entre la route et la place*** ; une fois le contraste détecté, nous allons pouvoir calculer la profondeur de la place grâce au capteur IR ainsi que le capteur de vitesse présent sur le disque du moteur. Pour ce faire, nous allons mesurer la vitesse et le temps passé à côté de la place.

Avec le ***temps*** et la ***vitesse*** nous allons pouvoir ***déterminer la taille et la profondeur.***

**Le code**

#Include

****

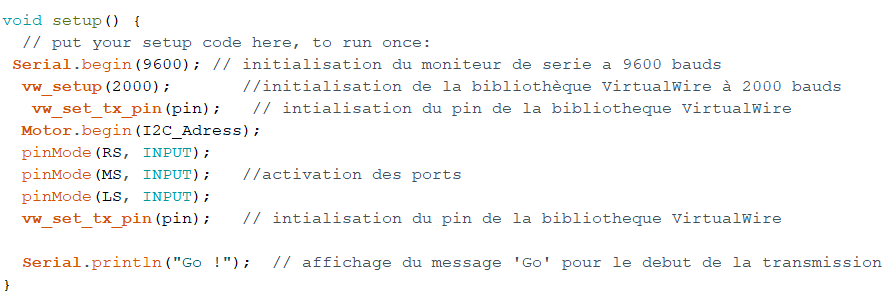
Le #include « Grove\_I2CMotor\_Driver.h » permet d’inclure la bibliothèque pour piloter le moteur I2C dans notre code.

Le #include <VirtualWire.h> est l’adresse matériel par défaut de la commande.

Le #DEFINE I2C\_Adress 0x0f permet de définir les constantes

Le int num permet de définir les constantes entrées

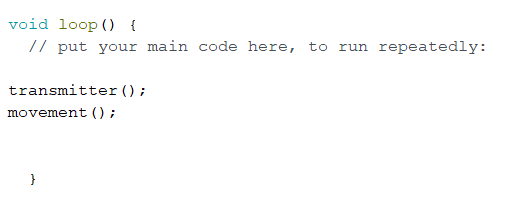
Void setup ()



Void setup permet ***d’initialiser toutes la valeurs et paramètres utilisés dans le programme Arduino***.

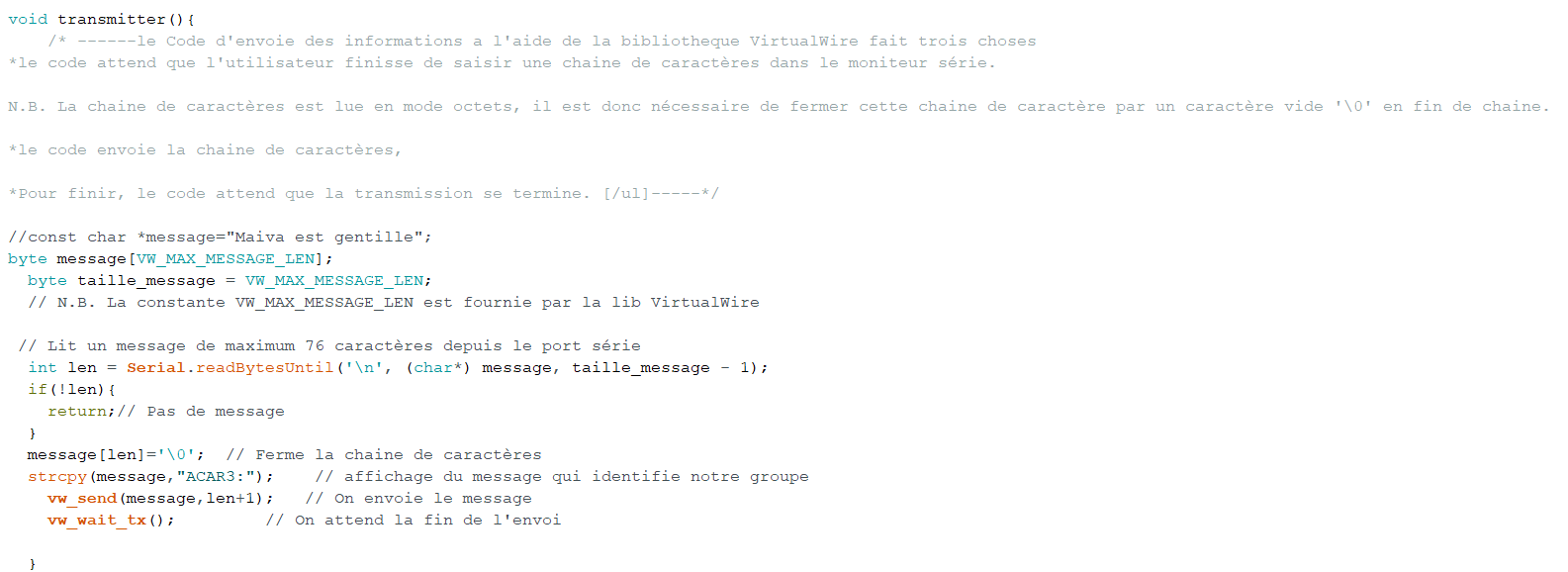
Serial.begin(2000) effectue une initialisation de la ***rapidité de modulation*** (en bauds) à 2000.

Void loop ()



Void loop est la partie du programme qui ***va se répéter à l’infini dans l’Arduino***.

Void transmitter ()



Void transmitter permet au code d’envoyer des informations à l’aide de la bibliothèque VirtualWire (attend que l’utilisateur finisse de saisir une chaines de caractère dans le moniteur série ; le code envoie la chaine de caractères ; le code attend que la transmission se termine).

Byte message et byte taille\_message ont la constante VM\_MAX\_MASSAGE\_LEN qui est fourni par la librairie VirtualWire et lit un message de max 76 caractères depuis le port série.

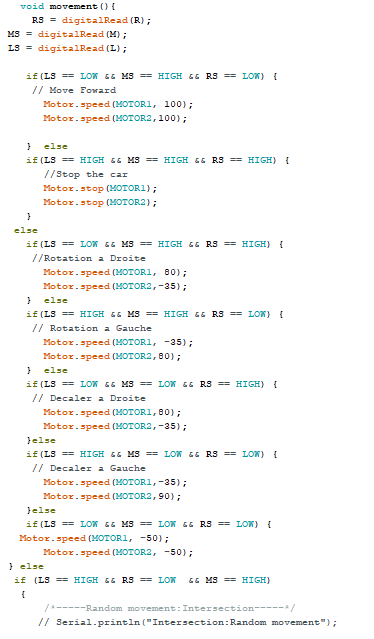
Message(len)=’\0’ ferme le chaines de caractères

Stropy(message, « ACAR3 : ») affiche le message qui identifie notre groupe

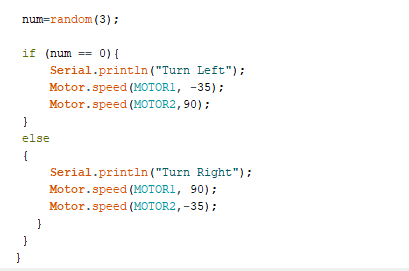
VW\_send(message, len+1) quand on envoi le message

VW\_wait\_tx () quand on attend la fin de l’envoi

Void movement ()



Void movement () permet de gérer les mouvements des moteurs ainsi que les informations des capteurs



Ceci par des conditions tels que if et else pour divers étapes dans les mouvements du robot.