RECEPTEUR

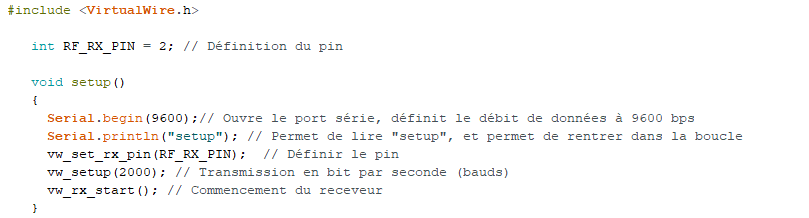
## INCLUDE

Tout d’abord nous ajoutons une nouvelle bibliothèque « ***Virtual Wire*** », celle-ci va nous permettre de permet ***d'envoyer et de recevoir des messages de moins de 77 octets***.

Ensuite nous définissons le pin que nous utilisons pour le transmetteur.

## VOID SETUP

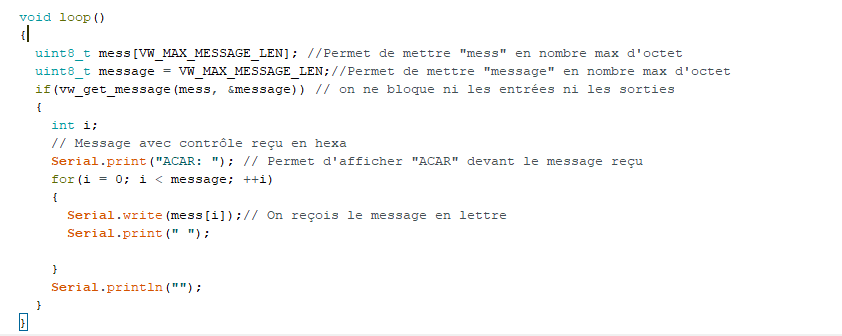
Le void setup va permettre de ***définir toutes les commandes qui vont nous aider à initialiser le ici receveur***, donc le ***port***, le ***pin***, la ***transmission en bauds***, et le ***start qui permet de lui demander de commencer à recevoir***.



## VOID LOOP

Pour la transmission du message, nous utilisons la commande « ***VW\_MAX\_MESSAGE\_LEN*** » à deux reprises ce qui nous permet de ***mettre les deux variables mess et message en nombre maximum d’octet lors pour qu’aucun message bloque lors de l’envoie***.

Ensuite nous utilisons un « ***if*** » pour ***ne pas bloquer ni les entrées ni les sorties***. La ligne qui suit est une ***création de la variable « i »*** (entier).

Pour finir, il suffit de lire les commentaires, donc ***nous recevons le message et on le contrôle***, le commentaire permet de dire que nous le ***recevons en hexadécimale mais nous l’avons modifié maintenant nous le recevons en lettre***. Donc nous ***écrivons un message devant le message que nous recevons « ACAR : »***. Et ensuite tous se joue dans le « Serial.write » car cette ligne nous a permis de modifier l’hexadécimal en lettre. 

EMETTEUR

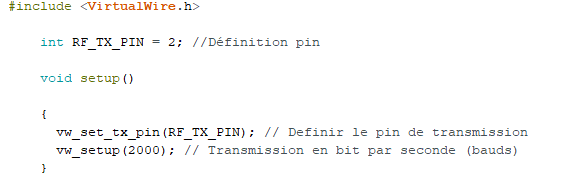
## INCLUDE

Tout d’abord nous ajoutons une ***nouvelle bibliothèque « VirtualWire*** ***»***, celle-ci va nous permettre de permet ***d'envoyer et de recevoir des messages de moins de 77 octets***.

Ensuite nous ***définissons le pin que nous utilisons pour le transmette*ur**.

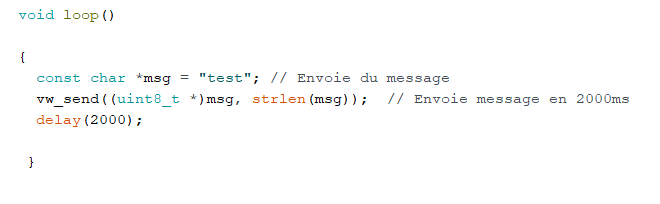
## VOID SETUP

Le void setup va permettre de ***définir le pin de transmission et d’initialiser la transmission en bit par seconde*** (bauds).



# VOID LOOP

C’est ici que passe l’envoie du message, donc comme on le voit si dessous dans le code dans le ***loop***, nous utilisons ***un constant caractère***, pour pouvoir envoyer plusieurs caractères et avoir simplement l’avoir en lecture seul. Donc ici le message envoyé sera « ***test*** », et le ***message sera envoyé en octet toutes les deux secondes***.

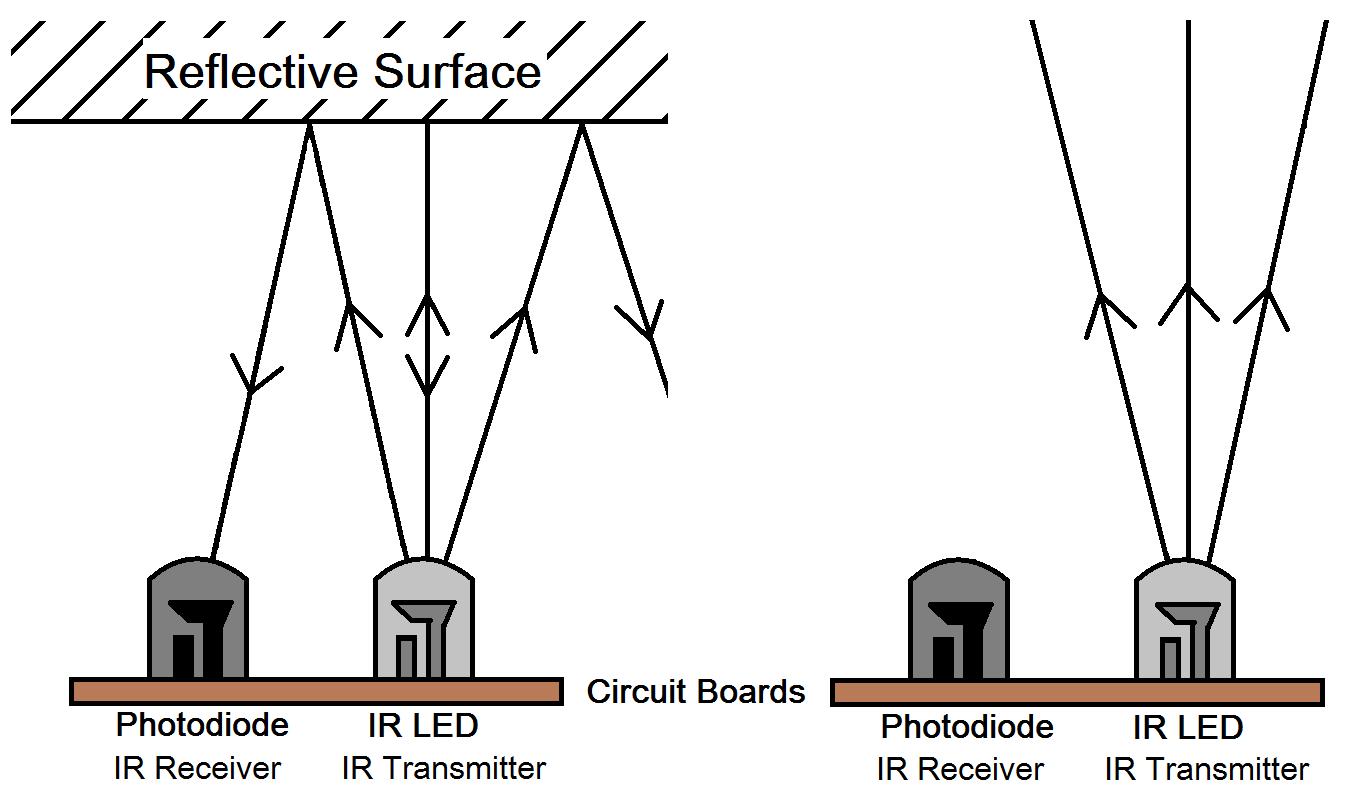


DOC Capteur IR

# Le capteur

## Fonctionnement

Comme son nom l’indique, ce capteur utilise un ***émetteur et un récepteur infrarouge***.



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 reçoit plus ou moins la lumière réfléchit***.

Selon la **couleur** de la surface ainsi que sa **distance**.

#### Exemple

Une surface blanche à 30 cm du capteur sera mieux qu’une surface verte à 30 cm également.

## 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***

Dans notre cas nous allons avoir besoin de détecter un contraste et de déterminer la « ***profondeur de ce contraste*** ».

#### Une image contenant horloge, objet Description générée avec un niveau de confiance très élevéNotre 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

## Une image contenant capture d’écran Description générée avec un niveau de confiance très élevéInclude

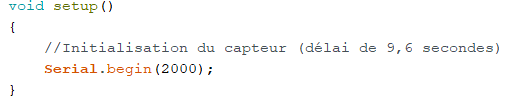
Le #define IR\_PROXIMITY\_SENSOR A1 permet de ***définir le pin*** sur lequel le capteur est branché.

Le #define ADC\_REF 5 permet de ***déterminer le voltage envoyé vers le capteur***, ici 5 Volts.

Cependant nous utilisons un Shield pour Arduino, la tension sera donc de 3,3 Volts

Float voltage est la variable qui ***va stocker le voltage envoyé par le capteur***. Cette variable va changer de valeur en fonction de la réflectivité du milieu.

## 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.

## Une image contenant capture d’écran Description générée avec un niveau de confiance très élevéVoid loop

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

## Voltage =getVoltage() permet d’obtenir les différentes valeurs envoyés par le capteur à un rythme régulier et non en continu.

Serial.print (‘’sensor voltage = ’’) permet ***d’afficher le message entre ‘’ ‘’***.

Ici on affiche « sensor voltage = »

Serial.print (voltage) permet ***d’afficher les valeurs envoyées par le capteur***.

Serial.print (‘’\n’’) permet de ***mettre un espace entre chaque ligne*** d’informations du capteur.

Delay(200) permet de mettre un ***délai de 200 millisecondes entre chaque affichage***.

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Float getVoltage() permet ***d’obtenir les valeurs du capteur de façon continue***.

Int sensor\_valeur permet d’obtenir une ***moyenne des sum*** envoyées par le capteur.  
La ***boucle for va demander 20 fois la valeur au capteur*** puis sensor\_value = sum/20 permet de ***faire une moyenne***.

On va finir par ***renvoyer le voltage au Serial.print*** pour ***l’afficher sur le moniteur Arduino***.