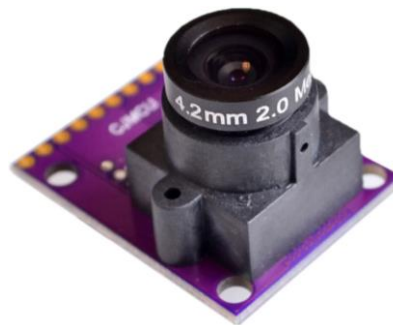
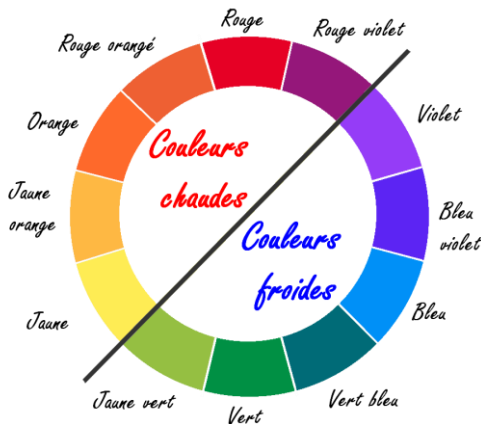
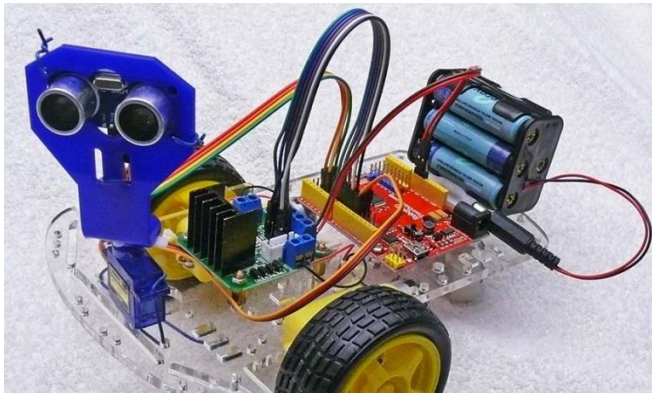


# Intelligence artificielle avec Arduino

Pascal MASSON  
(*pascal.masson@unice.fr*)



Version projection  
Edition 2023-2024-V15

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	1	0	0	1
8	0	0	0	0

image1

	1	2	3	4
1	1	1	1	1
2	1	0	0	1
3	1	0	0	1
4	1	1	1	1
5	1	0	0	1
6	1	0	0	1
7	0	0	0	0
8	0	0	0	0

image2

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	1	1	1	1
4	1	0	0	1
5	1	0	0	1
6	1	1	1	1
7	1	0	0	1
8	1	0	0	1

image3

## Introduction

### 1. Généralités sur l'IA

- ❑ Historique
- ❑ Les réseaux de neurones
- ❑ Exemples

### 2. Modélisation

- ❑ Cas d'un seul neurone : le perceptron
- ❑ Cas d'un seul neurone : l'apprentissage supervisé
- ❑ Cas d'un seul neurone : ligne de séparation
- ❑ Cas du « OU exclusif »

### 3. Couleurs chaudes ou froides

- ❑ Introduction
- ❑ Capteur et montage
- ❑ La librairie NeuralNetwork
- ❑ Mise en œuvre de la librairie NeuralNetwork
- ❑ Alors ? Couleurs chaudes ou froides ?

## 4. Evitement d'obstacles

- ❑ Introduction
- ❑ Configuration des capteurs
- ❑ Apprentissage du réseau de neurones
- ❑ Tests et corrections (réseau 3-6-2)

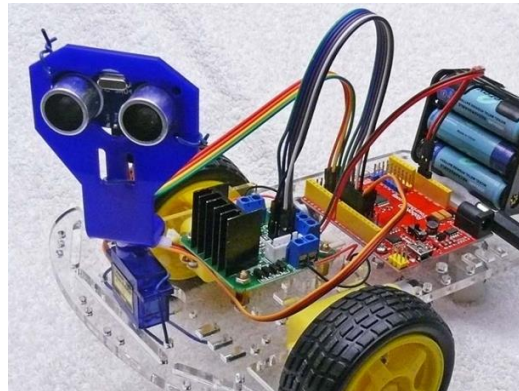
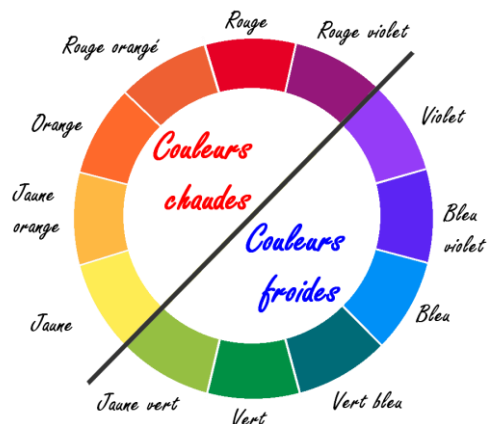
## 5. Reconnaissance de lettres

- ❑ Introduction
- ❑ Les contraintes
- ❑ Identification de la lettre « A »
- ❑ Capture d'images et identification

- Encyclopédie Larousse : «**L'intelligence artificielle (IA)** est l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence».
- L'IA ou encore « deep learning », apprentissage automatique, classification « machine learning » est quelque chose qui fascine (de par ses possibilités annoncées) mais qui effraie aussi (IA qui prend le contrôle du monde dans les livres/films de science-fiction).
- L'IA revient de façon cyclique sur le devant de la scène avec, ces dernières années, l'avantage de la puissance de calcul des ordinateurs modernes qui résulte de la course à l'intégration du transistor MOS.

- Ce cours se contentera d'illustrer l'apprentissage d'un petit réseau de neurones en adéquation avec les capacités de la carte arduino.
- Nous détaillerons pour cela 3 applications :

## Détection des couleurs chaudes ou froides



## Evitement d'obstacles pour un robot voiture

## Identification d'une lettre



## 1.1. Historique\*

### ❑ Gestation de l'IA (1943-1955)

- Travaux de McCulloch et Pitts qui ont introduit un modèle de neurone artificiel.
- Hebb propose une règle pour modifier les connections entre les neurones.
- Test de Turing : proposition de test d'IA fondée sur la faculté d'une machine à imiter la conversation humaine.

### ❑ Naissance de l'IA (1956)

- Conférence de 2 mois (!) sur ce thème est organisée par un petit groupe d'informaticiens.
- Le nom « Intelligence Artificielle » fut trouvé lors de cet évènement.

\*Référence : [www.electronique-mixte.fr/wp-content/uploads/2018/08/Cours-Intelligence-artificielle-15.pdf](http://www.electronique-mixte.fr/wp-content/uploads/2018/08/Cours-Intelligence-artificielle-15.pdf)

## 1.1. Historique

### ❑ Espoirs grandissants (1952-1969)

- Période très active.
- Un grand nombre de programmes furent développés pour résoudre des problèmes d'une grande diversité.

### ❑ Première déception (1966-1973)

- Les chercheurs en IA ont été trop optimistes.
- Exemple de la traduction automatique de texte qui nécessite la compréhension du texte.
- Les algorithmes développés sont très gourmands en mémoire et puissance de calculs.
- Arrêt des financements publics.

## 1.1. Historique

### ❑ Systèmes experts (1969-1979)

- Un système expert est un programme qui modélise la connaissance d'un expert.
- Le premier système expert s'appelle « DENDRAL » : le programme permettait d'identifier les constituants chimiques d'un matériau à partir de spectrométrie de masse et de résonance magnétique nucléaire.

### ❑ L'IA dans l'industrie (1980 ...)

- L'entreprise DEC utilise un système expert d'aide à la configuration de systèmes informatiques qui lui permet d'économiser des dizaines de millions de dollars par an.
- Entreprises qui forment leur propres équipes de recherche.
- USA et Japon financent de gros projets en IA.



## 1.1. Historique

### ❑ Le retour des réseaux de neurones (1986 ...)

- Découverte de la règle d'apprentissage dite de « back-propagation » qui permet aux réseaux de neurones d'apprendre des fonctions très complexes (règle déjà proposée en 1969 ...).
- C'est une des branches les plus actives de l'IA exploitée par exemple pour la fouille de données.

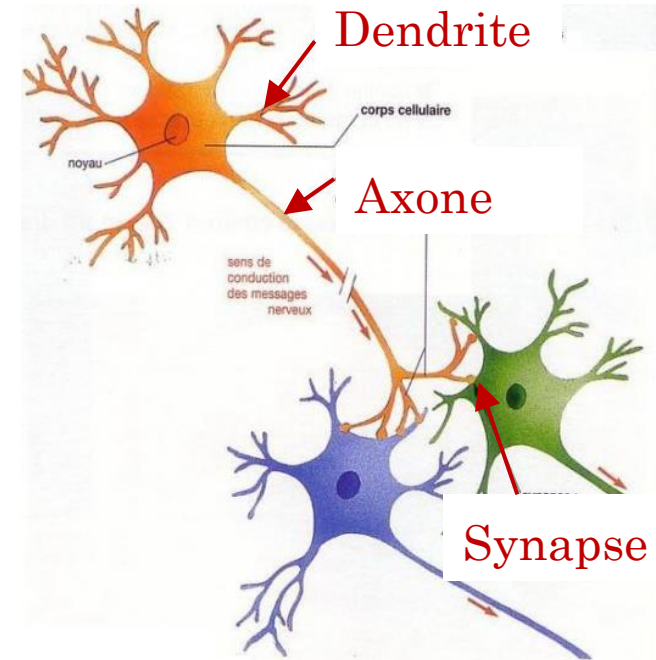
### ❑ L'IA moderne (1987 ...)

- La plupart des approches sont basées sur des théories mathématiques et des études expérimentales.
- 1997 : Deep Blue d'IBM bat le champion du monde d'échecs Garry Kasparov.
- L'IA est appliquée aux problèmes issus du monde réel.

## 1.2. Les réseaux de neurones

### □ En biologie

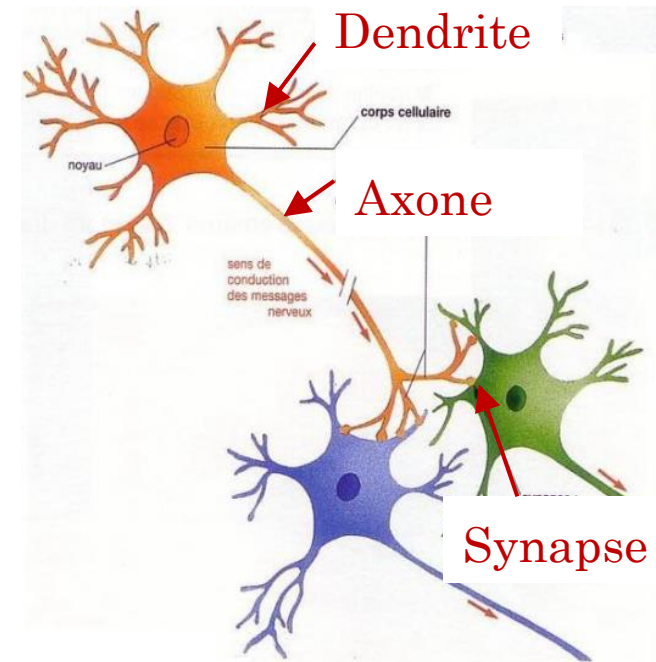
- Un neurone est une cellule du système nerveux spécialisée dans la communication et le traitement d'informations.
- Il se compose de très nombreuses ramifications de type dendritiques d'où proviennent les informations.
- L'axone permet de faire circuler l'information.
- Il se termine par de nombreuses ramifications et des synapses qui permettent le passage de l'information entre l'axone et les dendrites des neurones suivants.
- Le cerveau humain possède 100 milliards de neurones.



## 1.2. Les réseaux de neurones

### □ En biologie

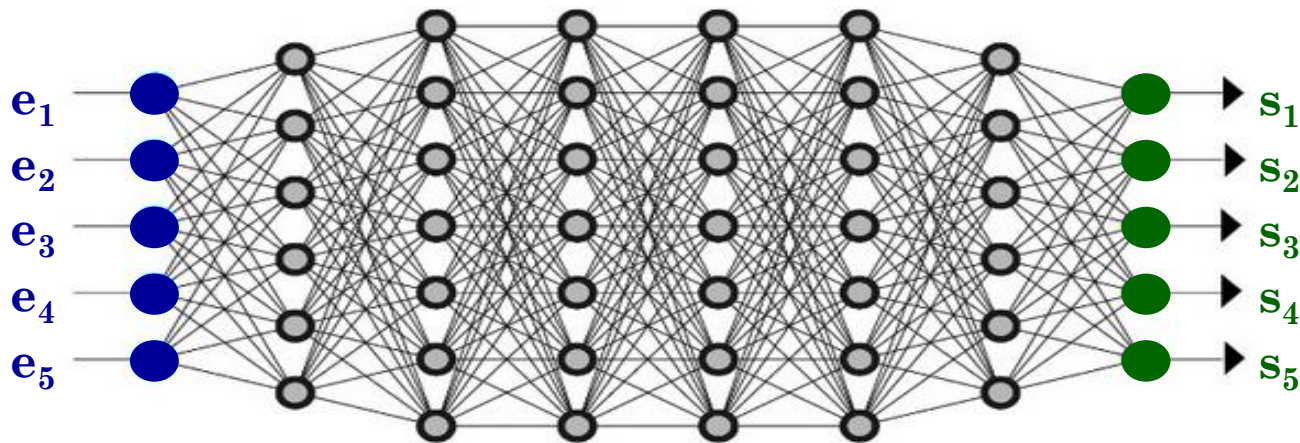
- Un neurone possède un seuil de déclenchement qui provoque la transmission d'une impulsion électrique le long de son axone.
- Ce seuil est atteint lorsque le potentiel membranaire atteint une certaine valeur.
- Les synapses en contact avec un neurone n'ont pas toutes le même « poids » donc la même influence sur l'atteinte du seuil de déclenchement.



## 1.2. Les réseaux de neurones

### □ En IA

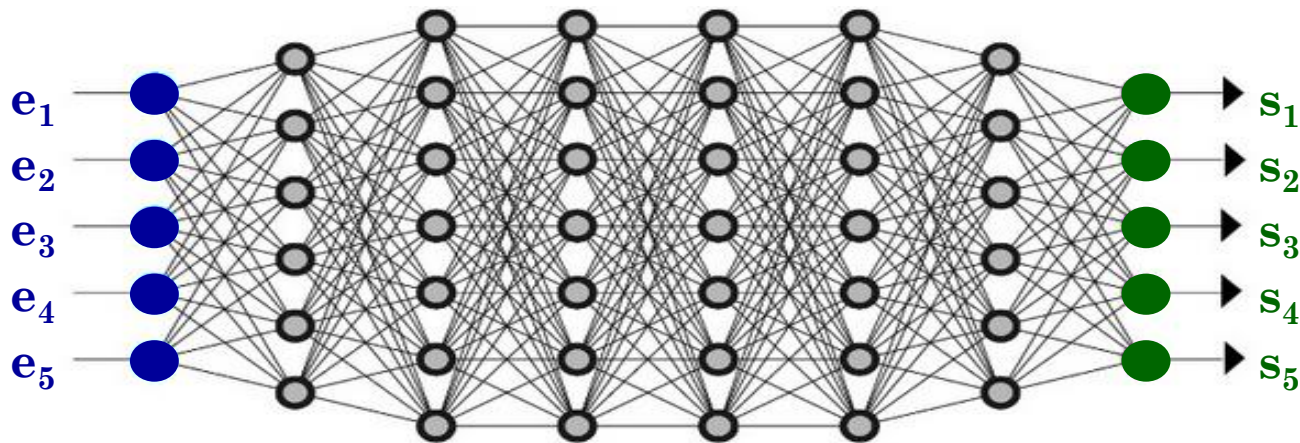
- En IA, un neurone s'appelle « perceptron »
- Les perceptrons sont agencés par couches (MLP : Multi Layer Perceptron) : la couche d'entrée, la couche de sortie et les couches cachées.
- L'influence d'un perceptron sur le suivant dépend de la valeur de poids (fils qui relie chaque perceptron) comme pour le neurone biologique.



## 1.2. Les réseaux de neurones

### □ En IA

- Les perceptrons ont aussi un seuil de déclenchement plus ou moins abrupt en fonction de la fonction choisie.
- L'apprentissage du réseau de neurones consiste à modifier la valeur des poids pour que la sortie corresponde à celle attendue.

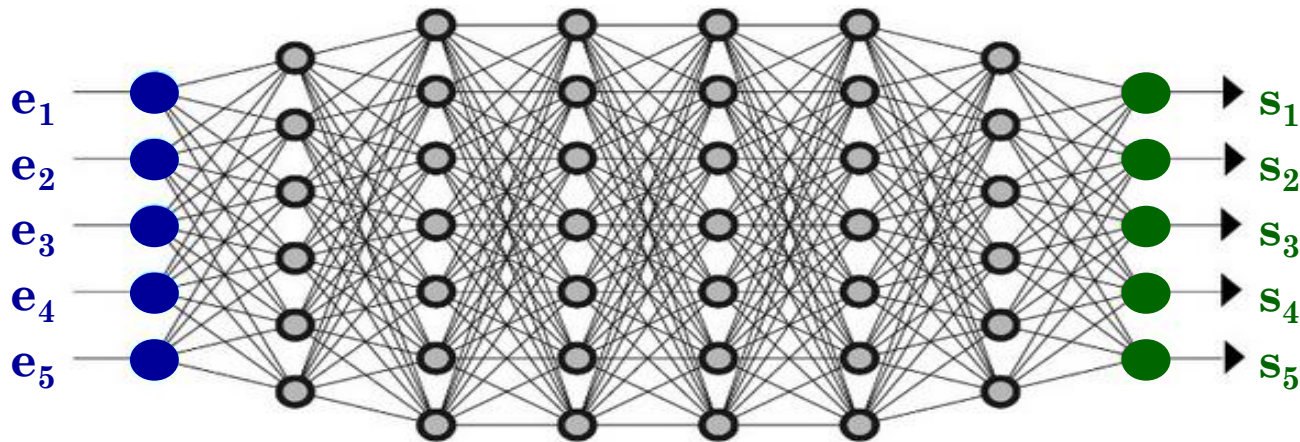




## 1.3. Exemples

### □ Réalisation de fonctions complexes

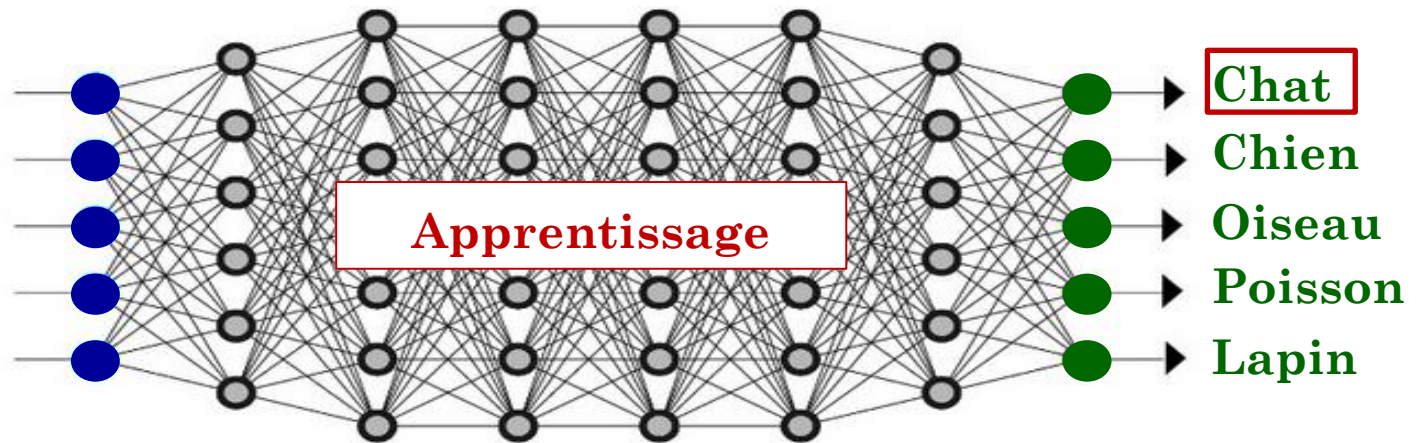
- Un réseau de neurone peut être configuré pour réaliser des fonctions complexes :  $s_1 = f_1(e_1, e_2, e_3, e_4, e_5)$ ,  $s_2 = f_2(e_1, e_2, e_3, e_4, e_5)$ , ...
- Une modification de l'apprentissage de ce réseau lui permettra de réaliser d'autres fonctions :  $s_1 = g_1(e_1, e_2, e_3, e_4, e_5)$ ,  $s_1 = g_2(e_1, e_2, e_3, e_4, e_5)$ , ...
- Si le réseau de neurones ne connaît pas une des combinaisons d'entrée/sortie, il donnera des valeurs de sortie qui dépendent des combinaisons qu'il connaît (base apprentissage).



## 1.3. Exemples

### □ Reconnaissance de formes

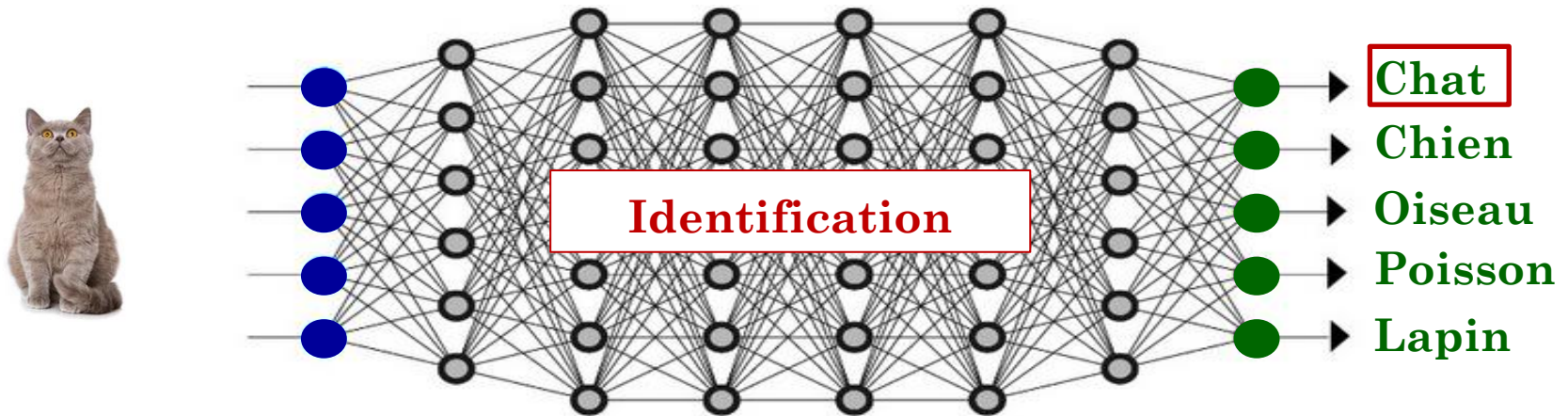
- Un réseau de neurones peut apprendre à reconnaître des formes.
- Il faut lui présenter un grand nombre de fois cette forme sous différents aspects, par exemple des images de chat et lui dire que c'est un chat.



## 1.3. Exemples

### □ Reconnaissance de formes

- Un réseau de neurones peut apprendre à reconnaître des formes.
- Il faut lui présenter un grand nombre de fois cette forme sous différents aspects, par exemple des images de chat et lui dire que c'est un chat.
- Si on lui présente une image d'un animal qui ressemble à un chat, le réseau doit alors indiquer que l'animal est un chat.

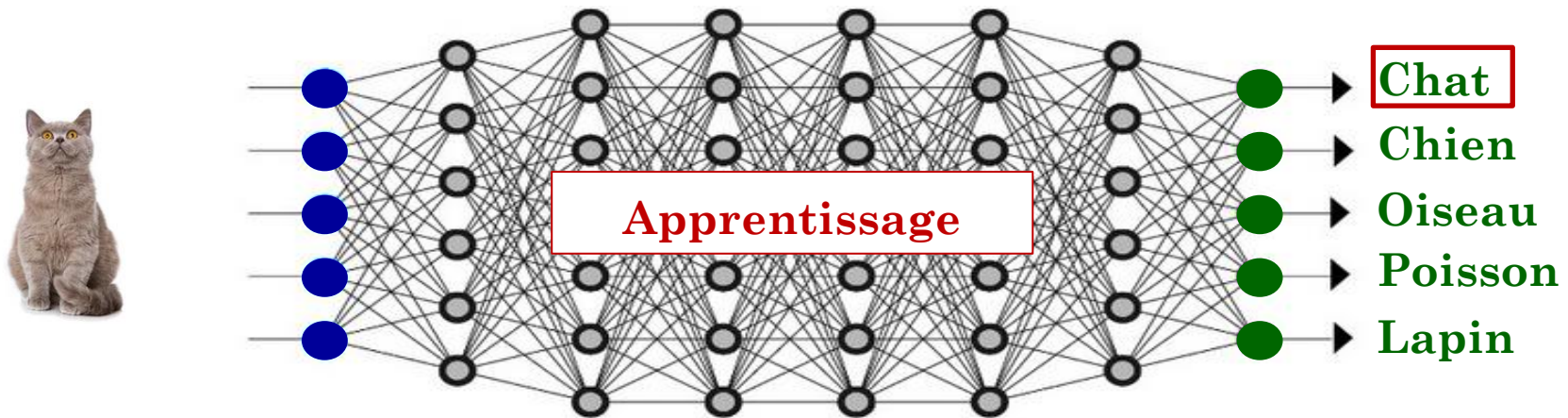




## 1.3. Exemples

### □ Reconnaissance de formes

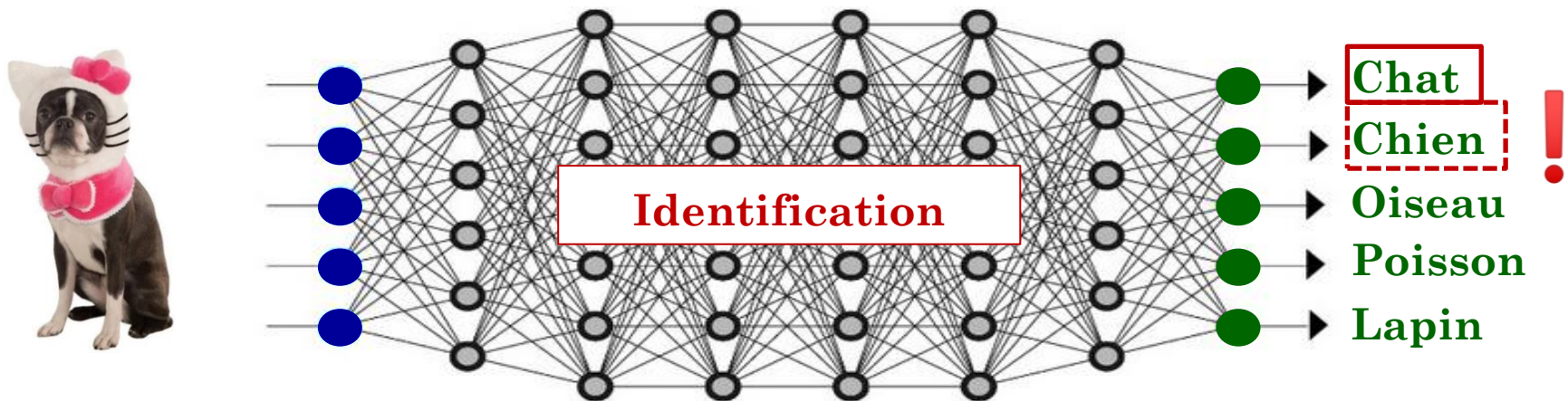
- Si le réseau ne reconnaît pas le chat, on peut ajouter l'image dans sa base d'apprentissage.
- Les poids sont alors modifiés.



## 1.3. Exemples

### □ Reconnaissance de formes

- Si le réseau ne reconnaît pas le chat, on peut ajouter l'image dans sa base d'apprentissage.
- Les poids sont alors modifiés .
- Le réseau peut aussi se tromper en identifiant comme étant un chat, un animal qui ne l'est pas.



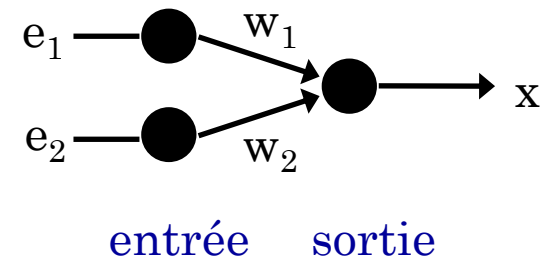
### 2.1. Cas d'un seul neurone : le perceptron

#### □ Fonctionnement

- Nous considérons un seul neurone en sortie et deux neurones en entrées notées  $e_1$  et  $e_2$ .
- L'influence de ces deux entrées sur le neurone dépend du poids que nous noterons  $w_1$  et  $w_2$ . Cela signifie que les deux entrées n'ont pas forcément la même influence sur le neurone.
- La valeur de la sortie, notée  $x$ , résulte de l'équation suivante :

$$x = f(a) = f(w_1 \times e_1 + w_2 \times e_2)$$

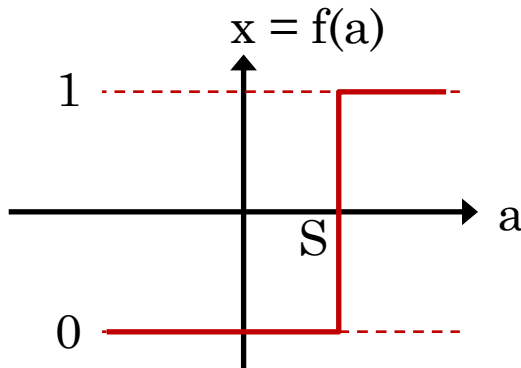
- $f(a)$  est une fonction de transfert



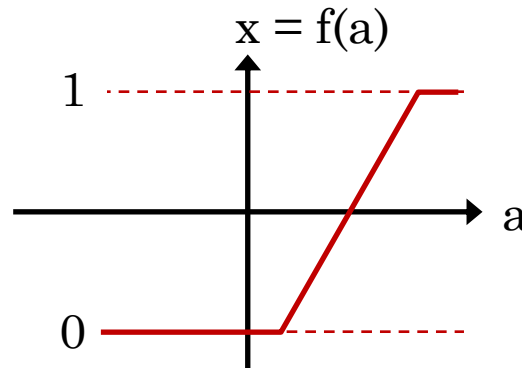
### 2.1. Cas d'un seul neurone : le perceptron

#### □ Fonctions de transfert

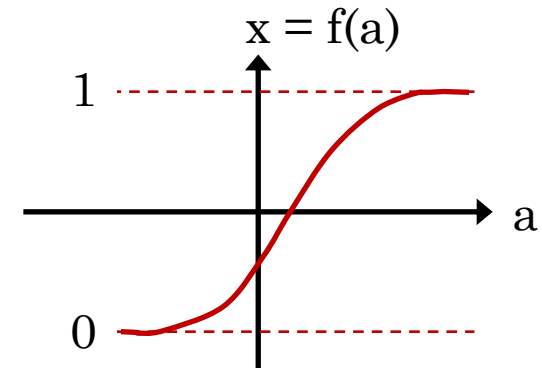
- Il existe une multitude de fonctions de transfert
- Elles peuvent être continues et ainsi permettre d'obtenir un intervalle continu de valeurs pour la sortie  $x$



**Fonction à seuil, S**



**Fonction linéaire  
par morceaux**

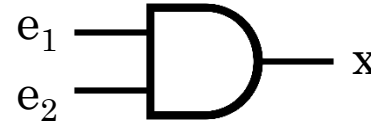


**Fonction sigmoïde**

## 2.1. Cas d'un seul neurone : le perceptron

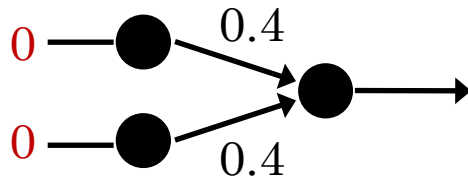
### □ Exemple de la porte « ET » à deux entrées

- Nous « rappelons » ici la table de vérité d'une telle porte en conservant la même notation que pour le perceptron mais avec une logique « 0 » et « 1 »

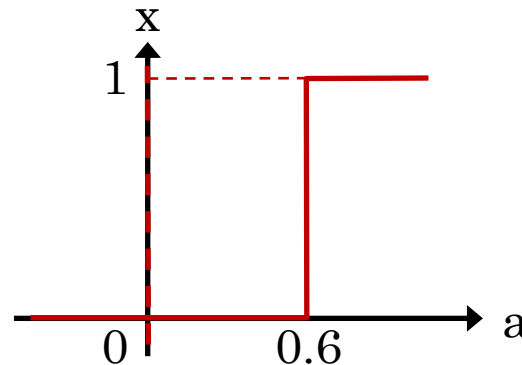


$e_1$	$e_2$	$x$
0	0	0
0	1	0
1	0	0
1	1	1

- Un choix judicieux des poids ( $w_1 = w_2 = 0.4$ ) ainsi que l'utilisation d'une fonction à seuil (avec  $S = 0.6$ ) permet d'obtenir le même comportement pour un perceptron



$$a = 0.4 \times 0 + 0.4 \times 0 = 0$$

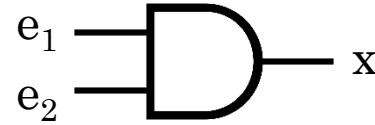


$e_1$	$e_2$	$x$
0	0	

### 2.1. Cas d'un seul neurone : le perceptron

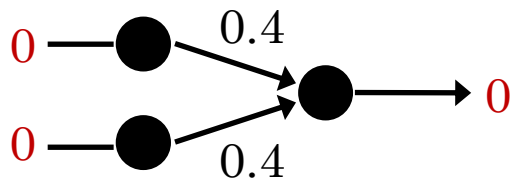
#### □ Exemple de la porte « ET » à deux entrées

- Nous « rappelons » ici la table de vérité d'une telle porte en conservant la même notation que pour le perceptron mais avec une logique « 0 » et « 1 »

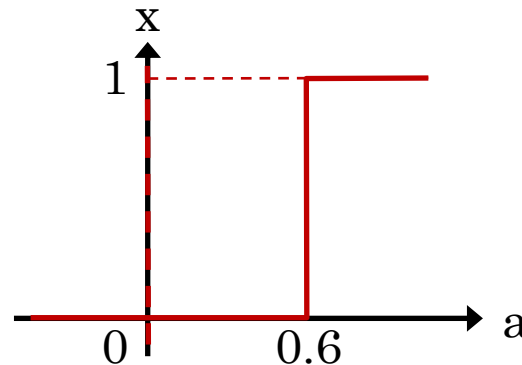


$e_1$	$e_2$	$x$
0	0	0
0	1	0
1	0	0
1	1	1

- Un choix judicieux des poids ( $w_1 = w_2 = 0.4$ ) ainsi que l'utilisation d'une fonction à seuil (avec  $S = 0.6$ ) permet d'obtenir le même comportement pour un perceptron



$$a = 0.4 \times 0 + 0.4 \times 0 = 0$$

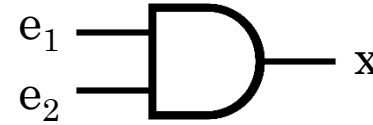


$e_1$	$e_2$	$x$
0	0	0

### 2.1. Cas d'un seul neurone : le perceptron

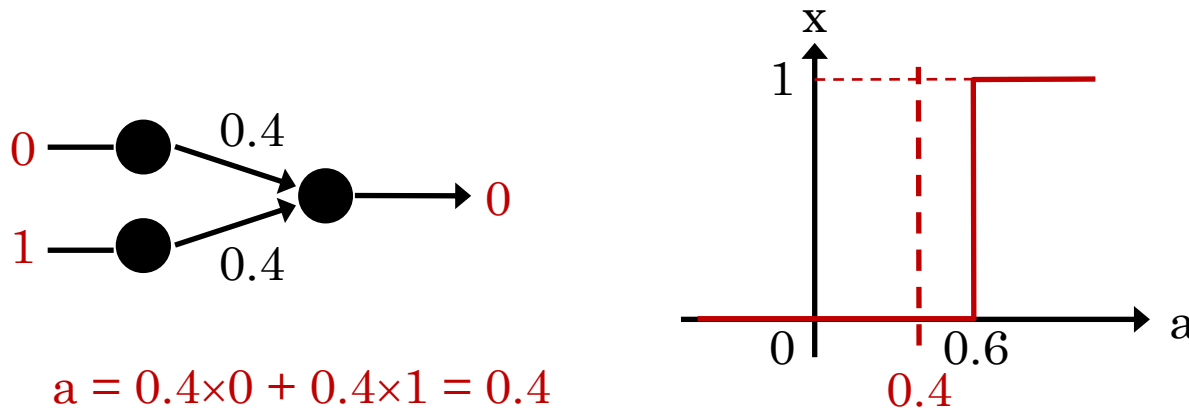
#### □ Exemple de la porte « ET » à deux entrées

- Nous « rappelons » ici la table de vérité d'une telle porte en conservant la même notation que pour le perceptron mais avec une logique « 0 » et « 1 »



$e_1$	$e_2$	$x$
0	0	0
0	1	0
1	0	0
1	1	1

- Un choix judicieux des poids ( $w_1 = w_2 = 0.4$ ) ainsi que l'utilisation d'une fonction à seuil (avec  $S = 0.6$ ) permet d'obtenir le même comportement pour un perceptron

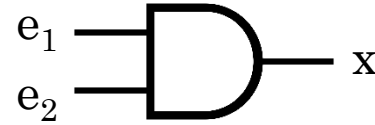


$e_1$	$e_2$	$x$
0	0	0
0	1	0

### 2.1. Cas d'un seul neurone : le perceptron

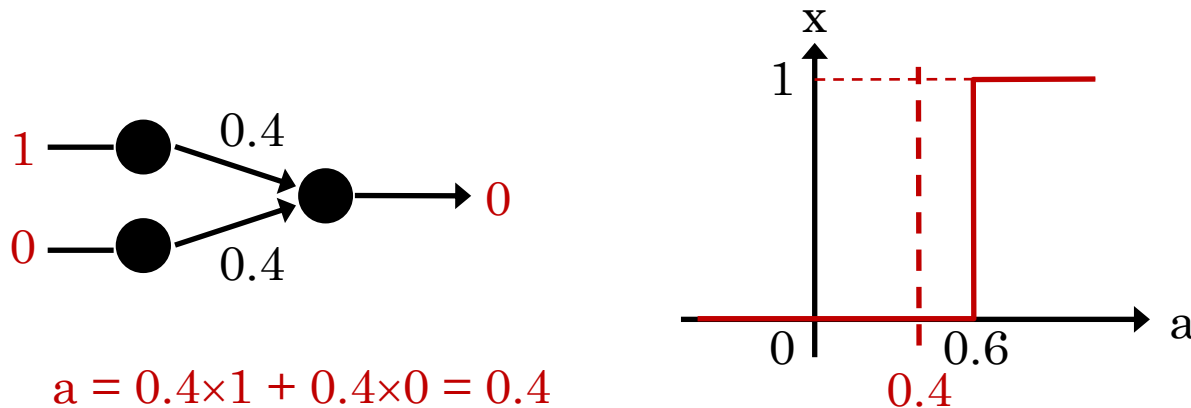
#### □ Exemple de la porte « ET » à deux entrées

- Nous « rappelons » ici la table de vérité d'une telle porte en conservant la même notation que pour le perceptron mais avec une logique « 0 » et « 1 »



$e_1$	$e_2$	$x$
0	0	0
0	1	0
1	0	0
1	1	1

- Un choix judicieux des poids ( $w_1 = w_2 = 0.4$ ) ainsi que l'utilisation d'une fonction à seuil (avec  $S = 0.6$ ) permet d'obtenir le même comportement pour un perceptron



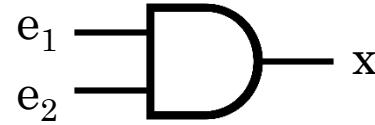
$e_1$	$e_2$	$x$
0	0	0
0	1	0
1	0	0



### 2.1. Cas d'un seul neurone : le perceptron

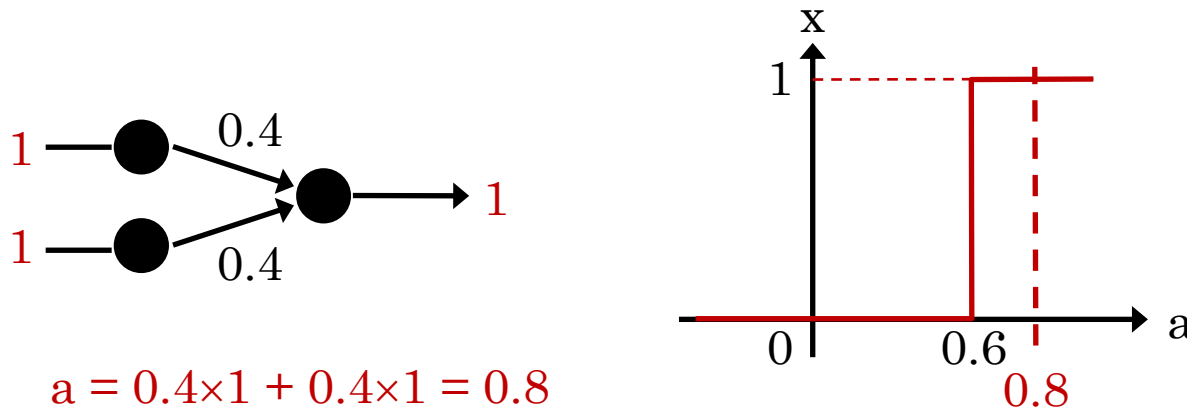
#### □ Exemple de la porte « ET » à deux entrées

- Nous « rappelons » ici la table de vérité d'une telle porte en conservant la même notation que pour le perceptron mais avec une logique « 0 » et « 1 »



$e_1$	$e_2$	$x$
0	0	0
0	1	0
1	0	0
1	1	1

- Un choix judicieux des poids ( $w_1 = w_2 = 0.4$ ) ainsi que l'utilisation d'une fonction à seuil (avec  $S = 0.6$ ) permet d'obtenir le même comportement pour un perceptron



$e_1$	$e_2$	$x$
0	0	0
0	1	0
1	0	0
1	1	1

### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

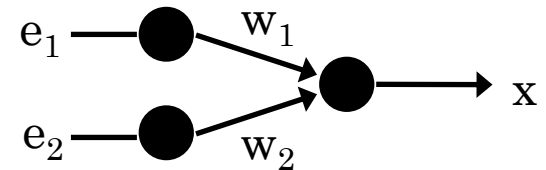
#### □ Introduction

- L'apprentissage est la partie la plus intéressante des réseaux de neurones
- Durant la phase d'apprentissage, les poids  $w$  sont modifiés jusqu'à obtenir le comportement souhaité
- Nous traitons ici le cas de l'apprentissage (règle de Hebb) qui prend en compte l'erreur observée en sortie

### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

#### ❑ Correction des poids : rétropropagation de l'erreur

- Nous conservons ici un exemple à 2 entrées qui peut être généralisé à n entrées
- Soit « d » le résultat qu'il faut obtenir quand on présente les valeurs d'apprentissage de  $e_1 \geq 0$  et  $e_2 \geq 0$  (une ligne de la base d'apprentissage).
- Soit x le résultat obtenu avec les poids
- On utilise une fonction à seuil (S) que l'on intègre dans le calcul de a :



$$a = w_1 \times e_1 + w_2 \times e_2 - S = \sum_{i=1}^2 (w_i \times e_i) - S$$

- La valeur de la sortie x dépend toujours de la fonction choisie :  $x = f(a)$
- Cependant, pour simplifier, nous considérerons que  $x = a$
- Donc pour  $e_i$  donné, une augmentation de  $w_i$  induit une augmentation de a

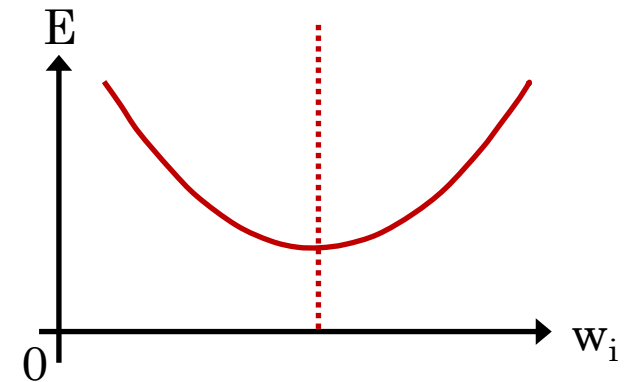
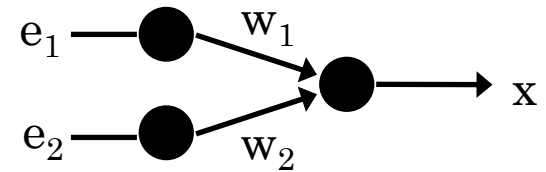
### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

#### □ Correction des poids : rétropropagation de l'erreur

- On définit l'erreur  $E$  comme étant la différence entre la sortie calculée et la sortie voulue :

$$E = \frac{1}{2}(d - x)^2 = \frac{1}{2}(d - a)^2 = \frac{1}{2}\left(d - \left(\sum_{i=1}^2 (w_i \times e_i) - S\right)\right)^2$$

- Pour une base d'apprentissage  $(e_1, e_2)$ , la courbe  $E = g(w_i)$  ( $w_1$  ou  $w_2$ ) a la forme d'une parabole qui présente de fait un minimum
- L'apprentissage correspond à la recherche de  $w_i$  qui permet de minimiser cette erreur



### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

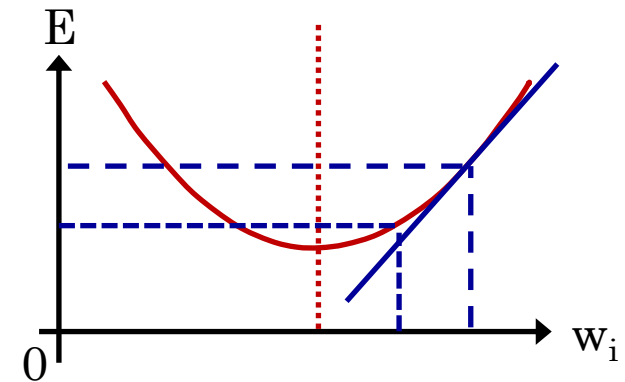
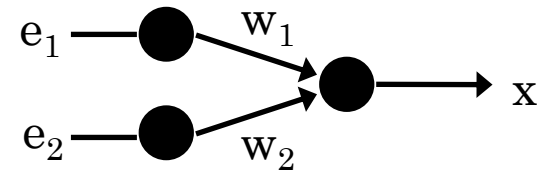
#### □ Correction des poids : rétropropagation de l'erreur

- On définit l'erreur  $E$  comme étant la différence entre la sortie calculée et la sortie voulue :

$$E = \frac{1}{2}(d - x)^2 = \frac{1}{2}(d - a)^2 = \frac{1}{2}\left(d - \left(\sum_{i=1}^2 (w_i \times e_i) - S\right)\right)^2$$

- Pour une base d'apprentissage  $(e_1, e_2)$ , la courbe  $E = g(w_i)$  ( $w_1$  ou  $w_2$ ) a la forme d'une parabole qui présente de fait un minimum
- L'apprentissage correspond à la recherche de  $w_i$  qui permet de minimiser cette erreur

- Le plus rapide pour rechercher ce minimum est d'utiliser la dérivée (pente) de la courbe (comme pour la méthode Newton-Raphson). Cela s'appelle la technique du gradient ou rétropropagation de l'erreur.



### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

#### □ Correction des poids : rétropropagation de l'erreur

- La pente de la courbe  $E = g(w_i)$  s'écrit :

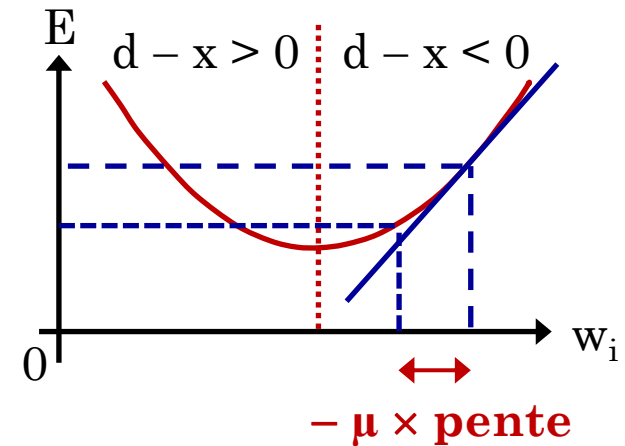
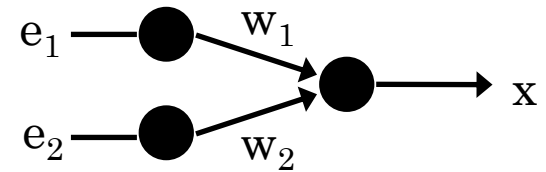
$$pente = \frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial x} \frac{\partial x}{\partial w_i} = \frac{\frac{\partial}{\partial x} \frac{1}{2}(d-x)^2}{\frac{\partial}{\partial x}} \frac{\partial \sum_{i=1}^2 (w_i \times e_i) - s}{\partial w_i}$$

$$pente = -(d - x) \times e_i$$

- Si  $(d - x) < 0$  cela signifie que l'action des entrées a trop d'impact sur la sortie  $x$ . Dans ce cas, les valeurs de  $w_1$  et  $w_2$  doivent être diminuées :

$$w_i = w_i + \mu \times (d - x) \times e_i$$

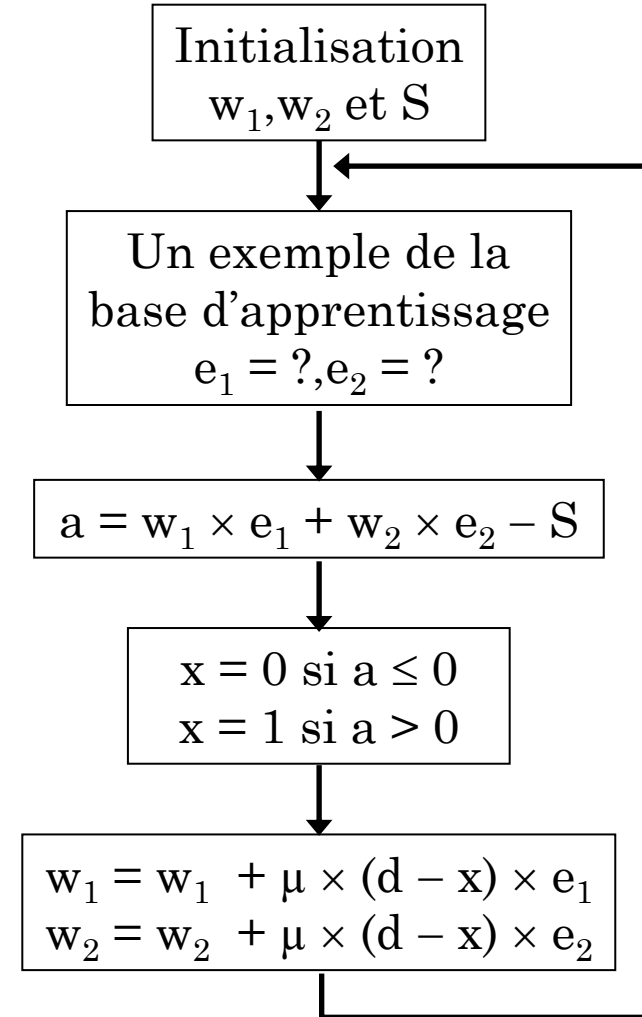
- $\mu (> 0)$  permet une modification plus ou moins importante des poids à chaque itération. Une valeur de  $\mu$  trop grande risque de faire osciller la valeur des poids. Une valeur de  $\mu$  trop faible augmente considérablement le nombre de boucles.



### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

#### □ Algorithme d'apprentissage

- L'algorithme ci-contre permet de déterminer la valeur des poids
- La boucle est exécutée tant que  $d$  est différent de  $x$  pour au moins une ligne de la base d'apprentissage.
- L'apprentissage est dit « supervisé » car il nécessite de connaître la réponse théorique (celle que le réseau doit apprendre)
- Au fur et à mesure des itérations, la valeur des poids est modifiée afin de réduire l'erreur ( $d - x$ ).



### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

#### □ Exemple de la porte « ET » à deux entrées

- Nous choisissons comme conditions initiales  $w_1 = w_2 = 0$  ainsi qu'un seuil  $S = 0.6$  et un pas de modification  $\mu = 0.5$
- On itère sur chaque ligne de la table de vérité jusqu'à obtenir 4 « Vrai » successifs

Table de vérité

e1	e2	d	Etat
0	0	0	1
0	1	0	2
1	0	0	3
1	1	1	4

		$a < 0$ donc $x = 0$	
W1 = 0	W2 = 0	Etat = 1	a = -0.6    x = 0    VRAI
W1 = 0	W2 = 0	← VRAI donc les poids ne sont pas modifiés	

$$a = w_1 \times e_1 + w_2 \times e_2 - S$$



### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

#### □ Exemple de la porte « ET » à deux entrées

- Nous choisissons comme conditions initiales  $w_1 = w_2 = 0$  ainsi qu'un seuil  $S = 0.6$  et un pas de modification  $\mu = 0.5$
- On itère sur chaque ligne de la table de vérité jusqu'à obtenir 4 « Vrai » successifs

Table de vérité

e1	e2	d	Etat
0	0	0	1
0	1	0	2
1	0	0	3

a < 0 donc x = 0

W1 = 0	W2 = 0	Etat = 1	a = -0.6	x = 0	VRAI
W1 = 0	W2 = 0	Etat = 2	a = -0.6	x = 0	VRAI
W1 = 0	W2 = 0	<div style="border-top: 2px solid red; width: 100%; height: 10px; position: relative;"><div style="position: absolute; left: 0; top: -5px; width: 10px; height: 10px; background: red; clip-path: polygon(50% 0%, 50% 100%, 0% 100%);"></div></div>			

← VRAI donc les poids ne sont pas modifiés

$$a = w_1 \times e_1 + w_2 \times e_2 - S$$

### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

#### □ Exemple de la porte « ET » à deux entrées

- Nous choisissons comme conditions initiales  $w_1 = w_2 = 0$  ainsi qu'un seuil  $S = 0.6$  et un pas de modification  $\mu = 0.5$
- On itère sur chaque ligne de la table de vérité jusqu'à obtenir 4 « Vrai » successifs

Table de vérité

e1	e2	d	Etat
0	0	0	1
0	1	0	2
1	0	0	3
1	1	1	4
0	0	0	1

$a < 0$ donc $x = 0$					
W1 = 0	W2 = 0	Etat = 1	a = -0.6	x = 0	VRAI
W1 = 0	W2 = 0	Etat = 2	a = -0.6	x = 0	VRAI
W1 = 0	W2 = 0	Etat = 3	a = -0.6	x = 0	VRAI
W1 = 0	W2 = 0	Etat = 4	a = -0.6	x = 0	FAUX
W1 = 0.5	W2 = 0.5				

FAUX donc les poids sont  
modifiés

$$\begin{aligned} w_1 &= w_1 + \mu \times (d - x) \times e_1 \\ w_2 &= w_2 + \mu \times (d - x) \times e_2 \end{aligned}$$

### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

#### □ Exemple de la porte « ET » à deux entrées

- Nous choisissons comme conditions initiales  $w_1 = w_2 = 0$  ainsi qu'un seuil  $S = 0.6$  et un pas de modification  $\mu = 0.5$
- On itère sur chaque ligne de la table de vérité jusqu'à obtenir 4 « Vrai » successifs

Table de vérité

e1	e2	d	Etat
0	0	0	1
0	1	0	2
1	0	0	3
1	1	1	4
0	0	0	1
0	1	0	2
1	0	0	3
1	1	1	4

W1 = 0	W2 = 0	Etat = 1	a = -0.6	x = 0	VRAI
W1 = 0	W2 = 0	Etat = 2	a = -0.6	x = 0	VRAI
W1 = 0	W2 = 0	Etat = 3	a = -0.6	x = 0	VRAI
W1 = 0	W2 = 0	Etat = 4	a = -0.6	x = 0	FAUX
W1 = 0.5	W2 = 0.5	Etat = 5	a = -0.6	x = 0	VRAI
W1 = 0.5	W2 = 0.5	Etat = 6	a = -0.1	x = 0	VRAI
W1 = 0.5	W2 = 0.5	Etat = 7	a = -0.1	x = 0	VRAI
W1 = 0.5	W2 = 0.5	Etat = 8	a = 0.4	x = 1	VRAI

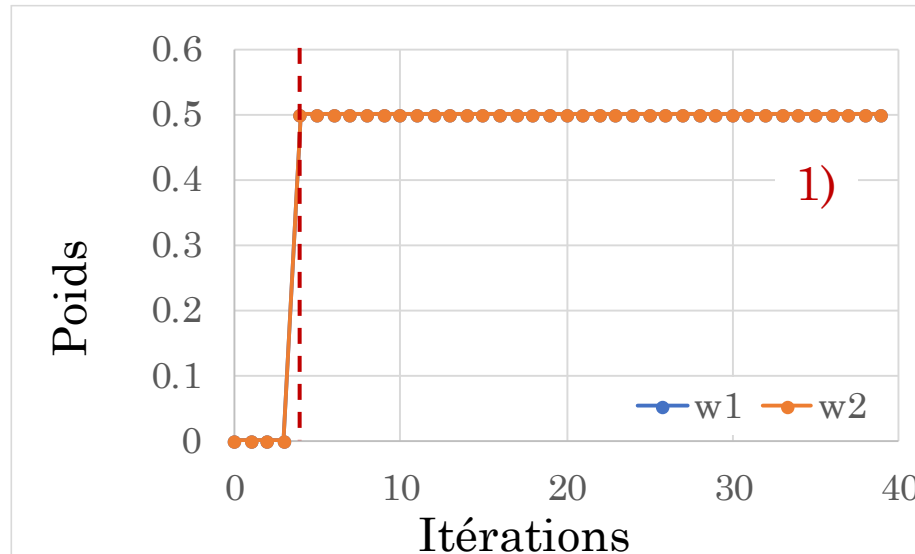
$$a = w_1 \times e_1 + w_2 \times e_2 - S$$

$a > 0$  donc  $x = 1$

### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

#### □ Exemple de la porte « ET » à deux entrées

- Le choix des valeurs de  $w_1$ ,  $w_2$ ,  $S$  et  $\mu$  peuvent rendre plus ou moins problématiques la recherche des poids  $w_1$ ,  $w_2$ . Le système peut même être oscillant



$$w_1 = 0$$

$$w_2 = 0$$

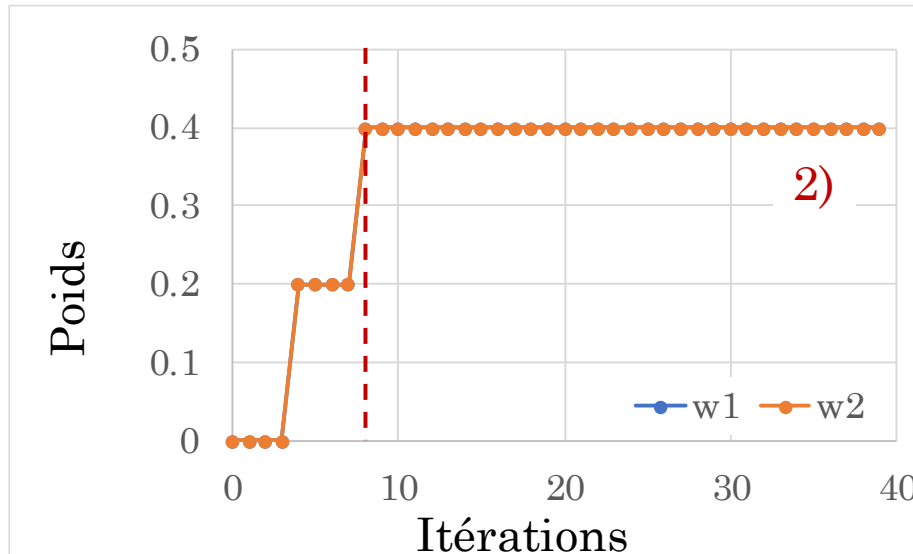
$$S = 0.6$$

$$\mu = 0.5$$

### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

#### □ Exemple de la porte « ET » à deux entrées

- Le choix des valeurs de  $w_1$ ,  $w_2$ ,  $S$  et  $\mu$  peuvent rendre plus ou moins problématiques la recherche des poids  $w_1$ ,  $w_2$ . Le système peut même être oscillant



$$w_1 = 0$$

$$w_2 = 0$$

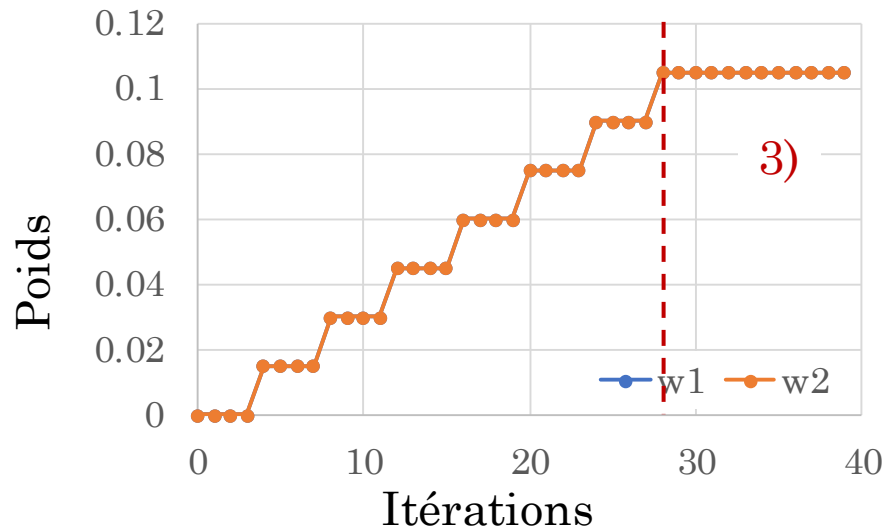
$$S = 0.6$$

$$\mu = 0.2$$

### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

#### □ Exemple de la porte « ET » à deux entrées

- Le choix des valeurs de  $w_1$ ,  $w_2$ ,  $S$  et  $\mu$  peuvent rendre plus ou moins problématiques la recherche des poids  $w_1$ ,  $w_2$ . Le système peut même être oscillant



$$w_1 = 0$$

$$w_2 = 0$$

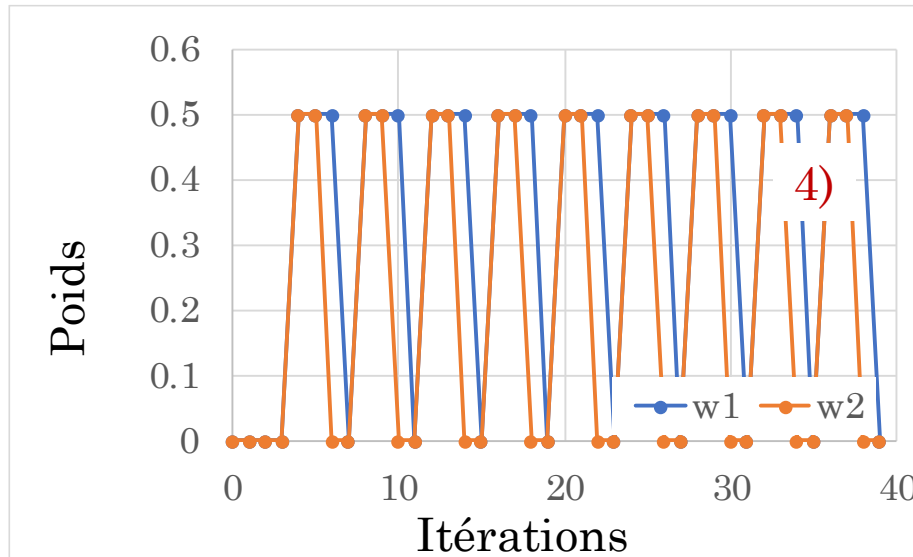
$$S = 0.2$$

$$\mu = 0.015$$

### 2.2. Cas d'un seul neurone : l'apprentissage supervisé

#### □ Exemple de la porte « ET » à deux entrées

- Le choix des valeurs de  $w_1$ ,  $w_2$ ,  $S$  et  $\mu$  peuvent rendre plus ou moins problématiques la recherche des poids  $w_1$ ,  $w_2$ . Le système peut même être oscillant



$$w_1 = 0$$

$$w_2 = 0$$

$$S = 0.2$$

$$\mu = 0.5$$

### 2.3. Cas d'un seul neurone : ligne de séparation

#### □ Mise en équation pour la porte « ET » à deux entrées

- Rappelons l'équation de « a » utilisée précédemment :

$$a = w_1 \times e_1 + w_2 \times e_2 - S$$

- Le basculement de « 0 » vers « 1 » de la sortie du perceptron se produit pour  $a > 0$ . En prenant  $a = 0$ , il est possible d'exprimer  $e_2$  en fonction de  $e_1$  :

$$e_2 = (S - w_1 \times e_1) / w_2$$

- Pour les valeurs utilisées (trouvées) précédemment cela amène à :

$$1) e_2 = (0.6 - 0.5 \times e_1) / 0.5 = 1.2 - e_1$$

$$2) e_2 = (0.6 - 0.4 \times e_1) / 0.4 = 1.5 - e_1$$

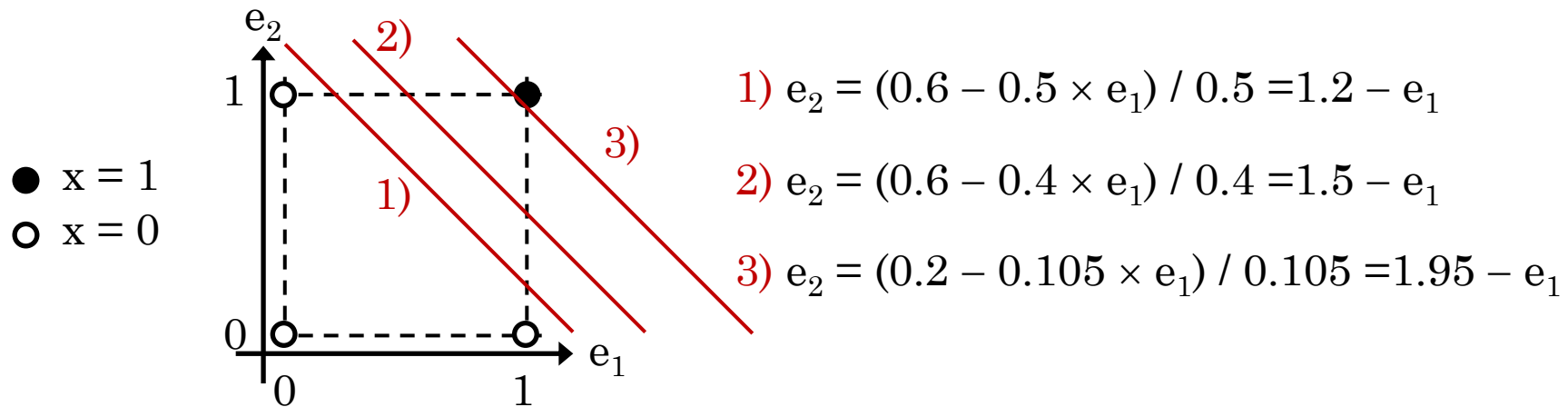
$$3) e_2 = (0.2 - 0.105 \times e_1) / 0.105 = 1.95 - e_1$$



### 2.3. Cas d'un seul neurone : ligne de séparation

#### □ Mise en équation pour la porte « ET » à deux entrées

- On peut représenter ces droites dans le plan  $(e_1, e_2)$

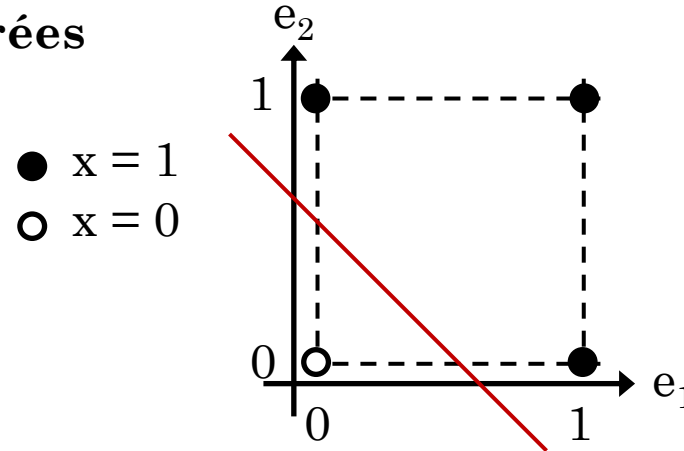


- On s'aperçoit que les droites séparent les 2 ensembles de points et qu'il existe une infinité de droites pouvant assurer cette séparation
- Il est cependant préférable de prendre une droite qui met un maximum de « distance » entre les 2 ensembles de points
- En conclusion, on peut dire que le perceptron permet de faire un classement linéaire des données (ensembles linéairement séparables)

### 2.3. Cas d'un seul neurone : ligne de séparation

#### □ La porte « OU » à deux entrées

- Pour cette porte, il existe aussi une infinité de droites pouvant séparer les deux ensembles de points



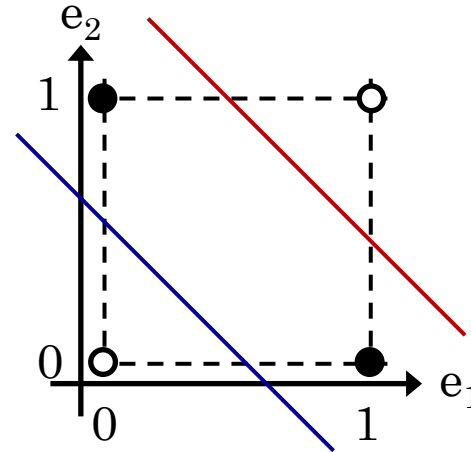
$e_1$	$e_2$	$x$
0	0	0
0	1	1
1	0	1
1	1	1

### 2.4. Cas du « OU exclusif »

#### □ Positionnement du problème

- Dans ce cas, on voit qu'il faut faire intervenir 2 droites pour séparer les 2 ensembles de points.

●  $x = 1$   
○  $x = 0$



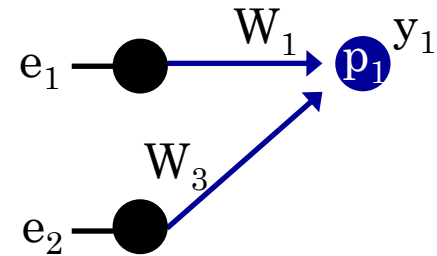
$e_1$	$e_2$	$x$
0	0	0
0	1	1
1	0	1
1	1	0

- Il est alors nécessaire de faire intervenir 3 couches de perceptrons : 1 en entrée (capteurs), 1 en sortie et 1 couche cachée.

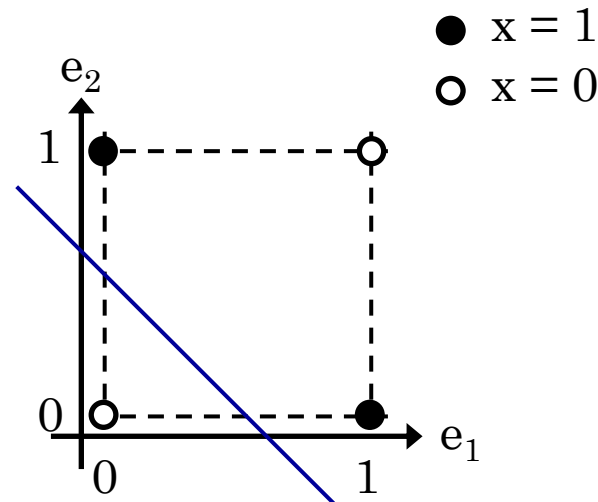
### 2.4. Cas du « OU exclusif »

#### □ Réseau de perceptrons

- Le perceptron  $P_1$  réalise la fonction « OU »



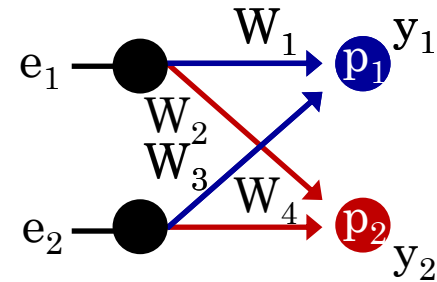
$e_1$	$e_2$	$y_1$		
0	0	0		
0	1	1		
1	0	1		
1	1	1		



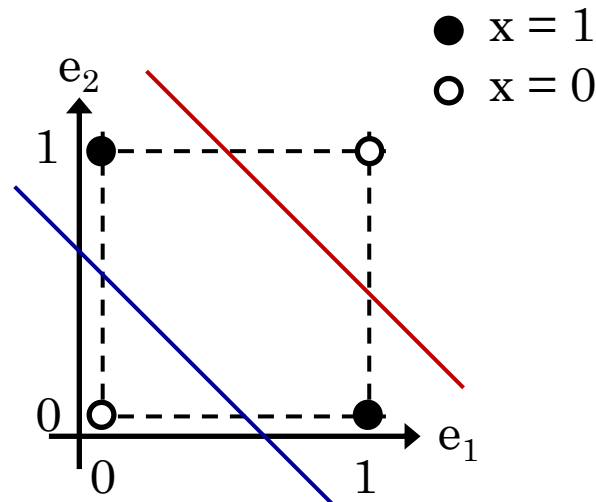
### 2.4. Cas du « OU exclusif »

#### □ Réseau de perceptrons

- Le perceptron  $P_1$  réalise la fonction « OU »
- Le perceptron  $P_2$  réalise la fonction « ET »
- Cela permet de supprimer la combinaison  $(y_1, y_2) = (0, 1)$



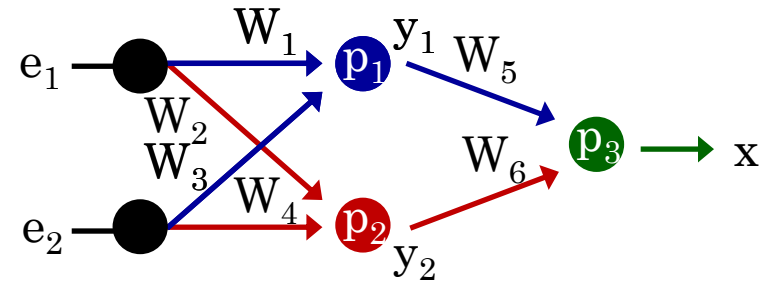
$e_1$	$e_2$	$y_1$	$y_2$	
0	0	0	0	
0	1	1	0	
1	0	1	0	
1	1	1	1	



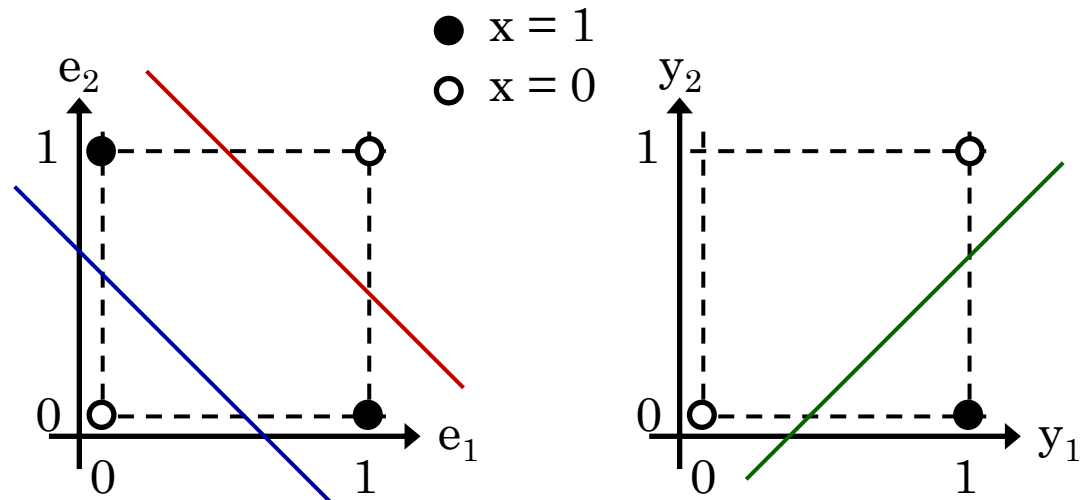
## 2.4. Cas du « OU exclusif »

### ❑ Réseau de perceptrons

- Le perceptron  $P_1$  réalise la fonction « OU »
- Le perceptron  $P_2$  réalise la fonction « ET »
- Cela permet de supprimer la combinaison  $(y_1, y_2) = (0, 1)$
- Il devient alors possible de séparer les 2 ensembles de points avec le perceptron de sortie



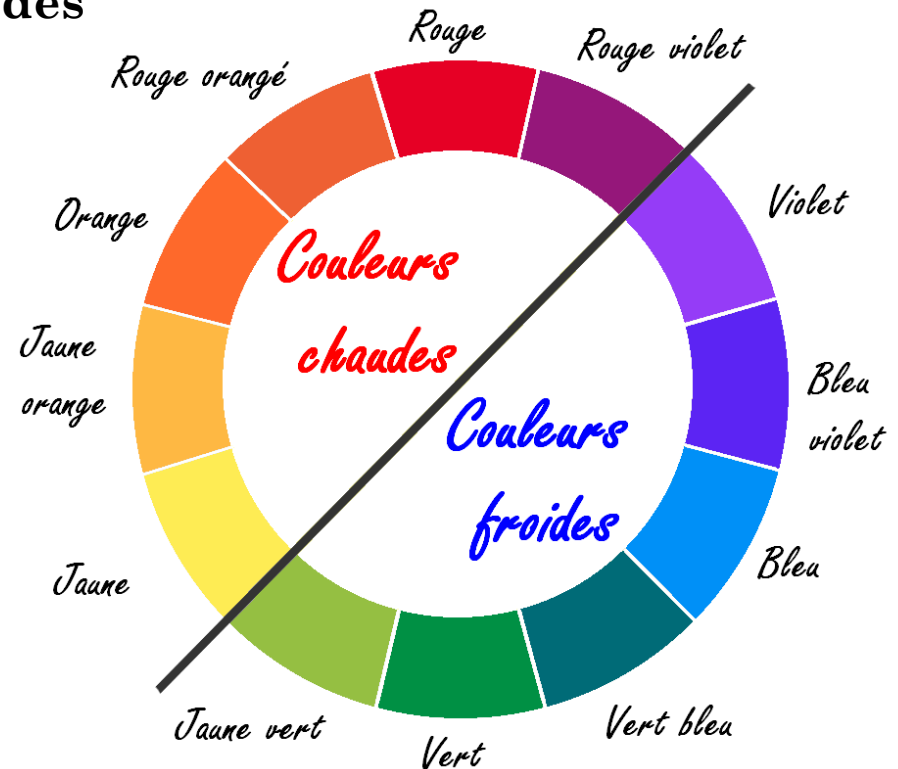
$e_1$	$e_2$	$y_1$	$y_2$	$x$
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0



## 3.1. Introduction

### □ Notion de couleurs chaudes et froides

- Les couleurs chaudes, font penser directement à la chaleur contrairement aux couleurs froides qui représentent plutôt la fraîcheur.
- L'orange est la couleur la plus chaude sur le cercle chromatique.
- En face, sa couleur complémentaire, le bleu, est quant à elle la couleur la plus froide.



## 3.1. Introduction

### □ Objectifs du chapitre

- Entraîner un petit réseau de neurones pour lui apprendre à déterminer si une couleur est chaude ou froide.
- Mettre en œuvre un capteur de couleur.

### □ Bibliographie

- L'exemple de ce chapitre a été fortement inspiré par :
  - ✓ Magazine Hackable n°31 (p. 60-79), [www.hackable.fr](http://www.hackable.fr)
  - ✓ [www.moretticb.com/blog/color-sensor-prototype-using-neural-networks/](http://www.moretticb.com/blog/color-sensor-prototype-using-neural-networks/)

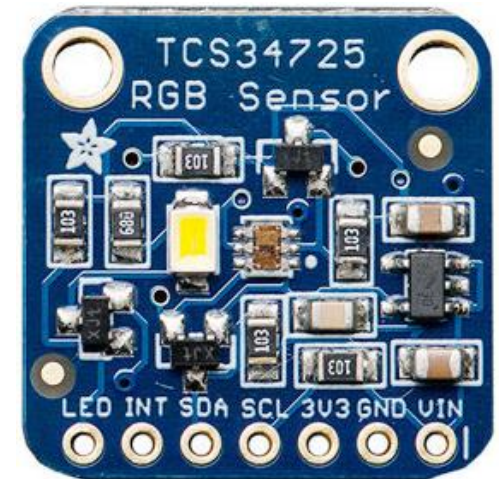




## 3.2. Capteur et montage

### □ Description du capteur TCS34725

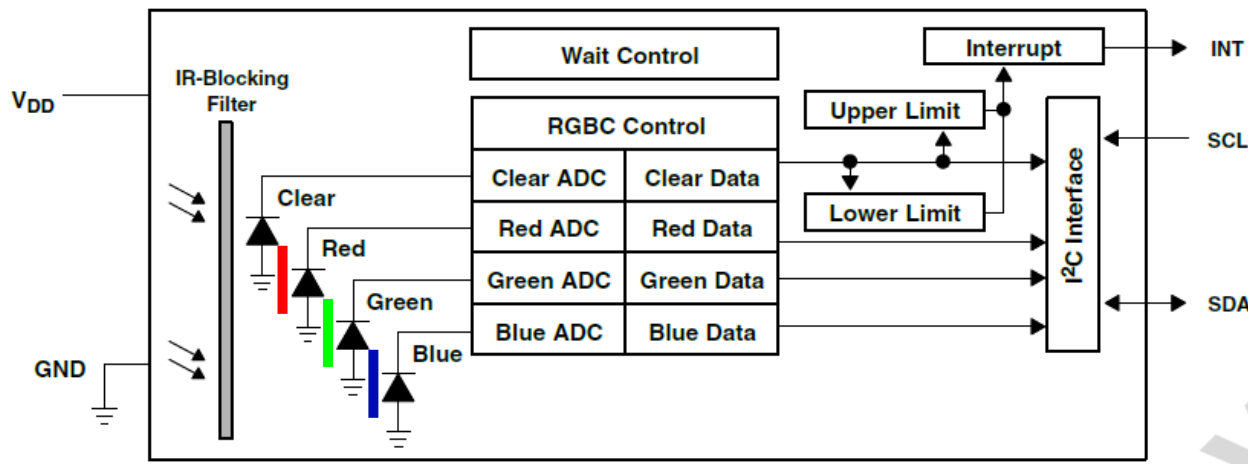
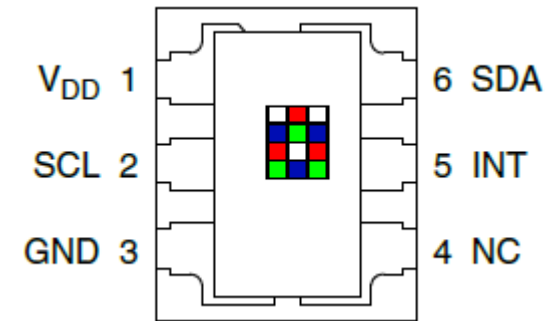
- Il existe plusieurs capteurs pour déterminer les composantes RGB d'une couleur. Ici, nous utiliserons le module TCS34725 d'Adafruit.
- Un régulateur de tension est intégré ce qui permet de l'alimenter en 5V (VIN-GND). Des levels-shifters sont présents sur les I/O.
- Les données transitent avec le bus I<sup>2</sup>C (SCL et SDA).
- Ce capteur renvoie les valeurs RGB d'une couleur.
- Le temps d'intégration de la mesure est réglable.
- Une LED (température 4150°K) permet d'éclairer l'objet à mesurer. La LED est éteinte lorsqu'on applique 0 V sur l'entrée LED.



## 3.2. Capteur et montage

### □ Description du capteur TCS34725

- La détermination des composantes RGB de la lumière est faite avec 9 photodiodes recouvertes d'une pellicule rouge ou vert ou bleu.
- 3 photodiodes sans filtre (clear) sont aussi présentes.
- Un filtre IR permet de ne pas parasiter la mesure avec les infra-rouges ambiants.



## 3.2. Capteur et montage

### □ Librairie

- La société Adafruit a développé une librairie pour ce capteur de couleur. Elle est téléchargeable ici (ou directement avec le manageur de librairie de l'IDE) :

[github.com/adafruit/Adafruit\\_TCS34725](https://github.com/adafruit/Adafruit_TCS34725)

### □ Fonctions

- Adafruit\_TCS34725 Nom = Adafruit\_TCS34725(X,Y) permet de déclarer le capteur avec un certain « Nom ». X correspond au temps d'intégration du signal et Y au gain.
- X peut être choisi parmi les valeurs ci-dessous. Le temps d'intégration peut être augmenté si la luminosité est faible.

TCS34725\_INTEGRATIONTIME\_2\_4MS, 24MS, 50MS, 101MS, 154MS, 700MS

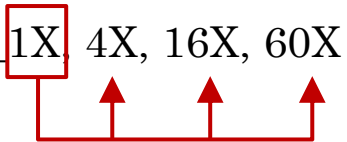


## 3.2. Capteur et montage

### □ Fonctions

- Y peut être choisi parmi les valeurs ci-dessous et permet d'ajuster la sensibilité du capteur.

TCS34725\_GAIN\_1X 4X, 16X, 60X

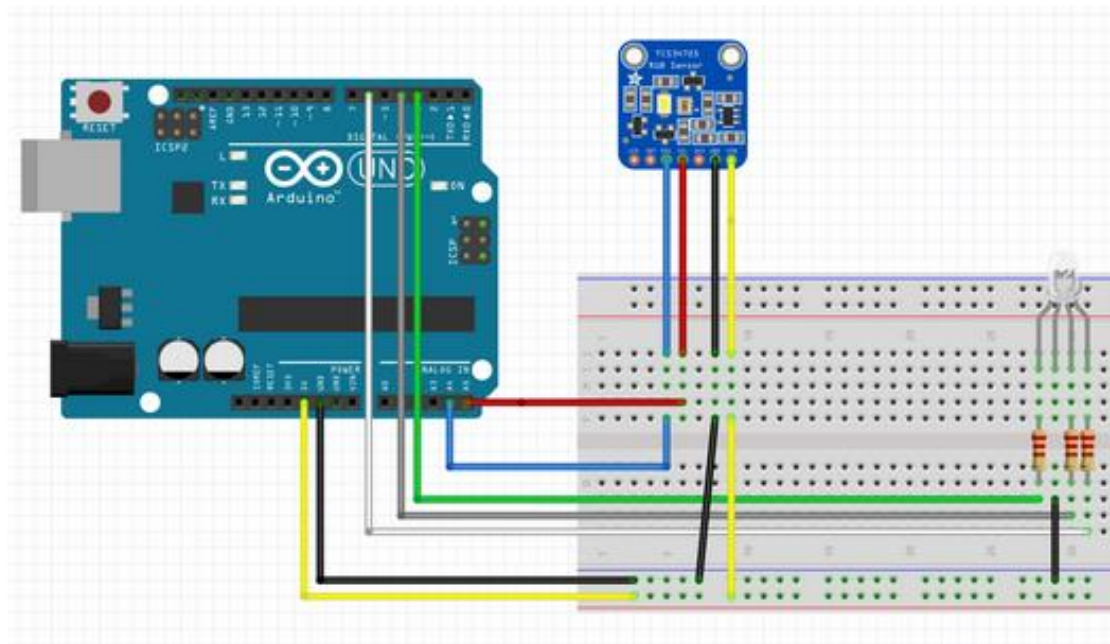


- Nom.**begin**() permet d'initialiser le capteur. Cette fonction renvoie « true » si le capteur est bien présent.
- Nom.getRawData(&r,&g,&b,&c) permet d'obtenir la valeur des composantes RGB. « c » pour « clear » (i.e. sans filtre RGB) correspond au total. r, g, b et c peuvent s'appeler autrement.

## 3.2. Capteur et montage

### □ Montage

- La communication se faisant via le bus I<sup>2</sup>C, le montage est très simple.
- On peut aussi ajouter une LED RGB qui permettra de visualiser la couleur qui a été mesurée.



## 3.2. Capteur et montage

### □ Initialisation du capteur et calibration

- Déclaration de la librairie et du protocole de communication « wire » pour le bus I<sup>2</sup>C :

```
#include <Wire.h>
#include "Adafruit_TCS34725.h"
```

- Déclaration du capteur que l'on appelle « tcs » avec un temps d'intégration de 50 ms et un gain de 4 :

```
Adafruit_TCS34725 tcs =
Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,TCS34725_
GAIN_4X);
```

- Initialisation du capteur (et de la communication série) :

```
void setup() {
  Serial.begin(9600);
  tcs.begin();
}
```

## 3.2. Capteur et montage

### □ Initialisation du capteur et calibration

- Pour utiliser au mieux ce capteur, il est nécessaire de faire une calibration en mesurant les valeurs RGB du blanc et du noir.
- Voici la procédure pour la mesure du blanc qui doit être mise dans le setup. La carte arduino attend de recevoir la lettre « y » pour sauver les valeurs et passer à la suite. En attendant, les valeurs RGB mesurées s'affichent sur le moniteur série.

```
Serial.println("Phase de calibration : identification du blanc. Entrer y  
quand c'est OK");  
while (Serial.read()!='y') {  
    tcs.getRawData(&rmax,&gmax,&bmax,&cmax);  
    Serial.print("R: "); Serial.print(rmax); Serial.print("\t");  
    Serial.print("G: "); Serial.print(gmax); Serial.print("\t");  
    Serial.print("B: "); Serial.print(bmax); Serial.print("\t");  
    Serial.print("C: "); Serial.println(cmax);  
    delay(1000); }
```

### 3.2. Capteur et montage

#### □ Mesures des couleurs

- Une fois le calibrage fait, on passe dans la « loop » et pour chaque mesure, il faut convertir les valeurs RGB en un nombre compris entre 0 et 255.
- Pour cela on utilise les valeurs min et max de RGB :

```
void loop() {  
    tcs.getRawData(&r,&g,&b,&c);  
    int rmap=map(r,rmin,rmax,0,255);  
    int gmap=map(g,gmin,gmax,0,255);  
    int bmap=map(b,bmin,bmax,0,255);
```

- Comme il est possible qu'une mesure (par exemple de r) sorte de l'intervalle [rmin,rmax], il faut contraindre le résultat à rester dans l'intervalle [0,255] :

```
int R = constrain(rmap,0,255);  
int G = constrain(gmap,0,255);  
int B = constrain(bmap,0,255);
```



### 3.2. Capteur et montage

#### □ Exemples de résultats

- La mesure d'une feuille blanche donne :

R: 4538 G: 4539 B: 4154



- La mesure d'un carré noir dessiné au feutre donne :

R: 365 G: 324 B: 319



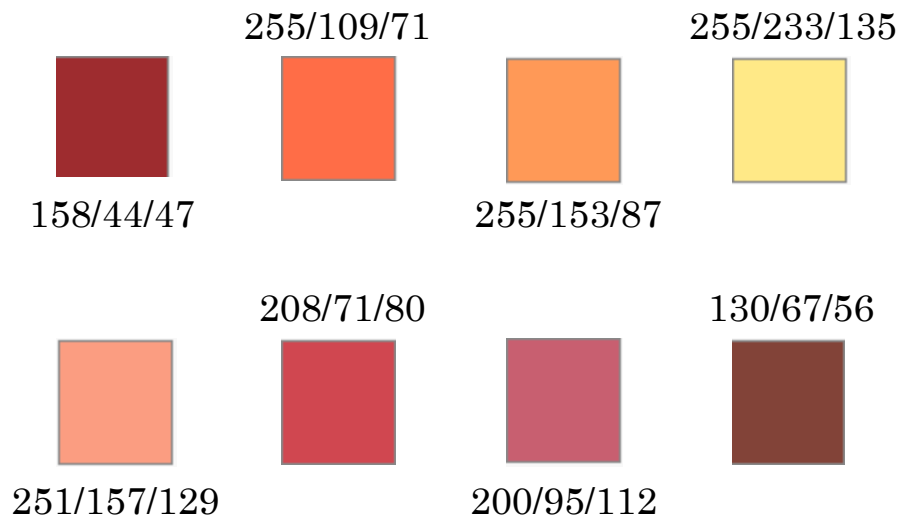
- Une fois ces 2 mesures effectuées, on passe à la détermination des composantes RGB d'autres couleurs.

## 3.2. Capteur et montage

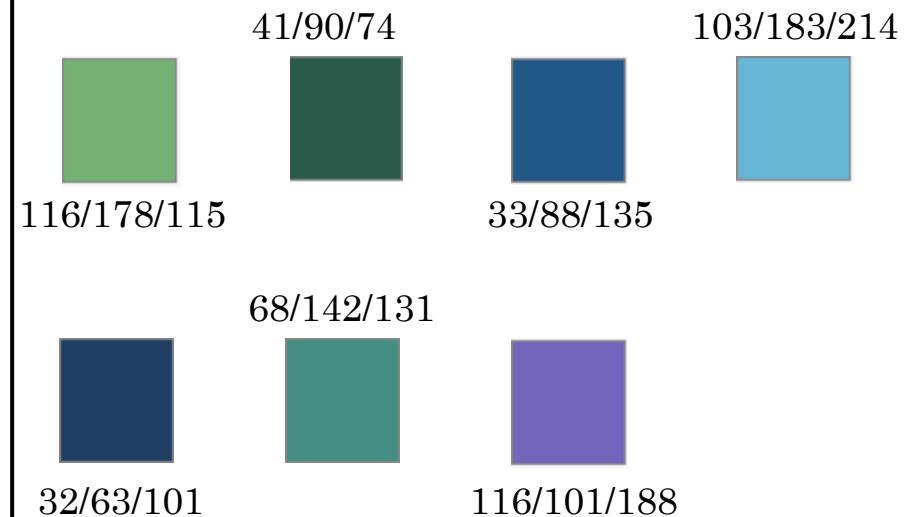
### □ Exemples de résultats

- Des carrés de couleur sont dessinés sur une feuille blanche (avec des feutres « empruntés » à mes enfants) puis mesurés par le capteur :

#### Couleurs chaudes



#### Couleurs froides



- Cette base servira par la suite pour déterminer (avec le réseau de neurones) si les couleurs sont chaudes ou froides.

### 3.3. La librairie NeuralNetwork

#### □ Présentation

- Le .zip de la librairie se trouve sur le site arduino :

[www.arduinelibraries.info/libraries/neural-network](http://www.arduinelibraries.info/libraries/neural-network)

- On peut aussi aller sur la page github de l'auteur de la librairie :

[github.com/GiorgosXou/NeuralNetworks](https://github.com/GiorgosXou/NeuralNetworks)

- Les spécifications annoncées sont :

- ✓ Algorithme optimisé pour une utilisation moindre de la SRAM.
- ✓ La fonction d'activation est de type sigmoïde.
- ✓ Stockage des données dans la mémoire FLASH et non la SRAM.
- ✓ Simplicité d'utilisation.
- ✓ Les entrées doivent avoir pour valeur 1 au maximum.

## 3.3. La librairie NeuralNetwork

### □ Les fonctions pour l'apprentissage

- **NeuralNetwork** Nom(couches,X) permet de définir un réseau de neurones avec un certain « Nom ». X correspond au nombre de couches (entrée + cachées + sortie). La variable « couches » est un tableau qui contient le nombre de neurones par couche. **Il faut mettre cette fonction avant le setup pour que le réseau soit reconnu dans tous les « void » du programme.**
- Il faut définir la base d'apprentissage avec :
  - ✓ Le tableau à 2 dimensions des entrées, `const float entrees [X1] [X2]`, où X1 correspond au nombre d'exemples pour l'apprentissage et X2 au nombre de neurones en entrée.
  - ✓ Le tableau à 2 dimensions des sorties attendues pour les entrées de la base, `const float sortiesAttendues [X1] [X3]`. X1 correspond au nombre d'exemples pour l'apprentissage et X3 correspond au nombre de neurones en sortie.

### 3.3. La librairie NeuralNetwork

#### □ Les fonctions pour l'apprentissage

- Nom. **FeedForward** (entrees) permet de calculer la sortie pour un exemple de la base d'apprentissage. Il est possible de sauver la valeur des sorties dans un tableau pour affichage.
- Nom. **BackProp** (sortiesAttendues) permet de corriger les poids et bias en fonction de l'erreur sur la sortie.
- Il faut utiliser successivement les 2 fonctions précédentes sur toute la base d'apprentissage et répéter l'opération un certain nombre de fois.
- Nom. **print** () permet d'imprimer tous les poids.

### 3.3. La librairie NeuralNetwork



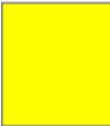
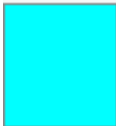

#### □ Les fonctions pour l'utilisation du réseau

- Il est aussi possible de définir un réseau de neurones et de lui donner la valeur des poids et des bias (constante S utilisée au chapitre 2) pour chaque couche
- **NeuralNetwork** Nom(couches,poids,bias,X) permet de définir un réseau de neurones avec un certain « Nom ». X correspond au nombre de couches et « couches » est un tableau qui contient le nombre de neurones par couche. « poids » est un tableau à une ligne qui donne dans l'ordre (entrée vers sortie) la valeur des poids. « bias » est un tableau à une ligne (X – 1) éléments qui donne la valeur des bias (pas de bias sur la couche d'entrée).
- Le calcul de la sortie se fait toujours avec la fonction Nom.**FeedForward** (entrees).

## 3.4. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- On utilise 6 couleurs pour la base d'apprentissage. L'intervalle  $[0,255]$  doit être ramené à  $[0,1]$
- La valeur « 1 » en sortie signifie que la couleur est chaude sinon la sortie vaut « 0 »

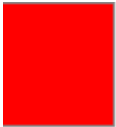


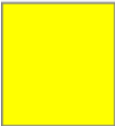
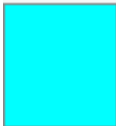
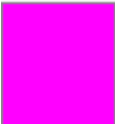
Rouge : 255/0/0		
Vert : 0/255/0		
Bleu : 0/0/255		
Jaune : 255/255/0		
Cyan : 0/255/255		
Magenta : 255/0/255		

```
const float entrees [6][3] = {
  {1.00,0.00,0.00},//rouge
  {0.00,1.00,0.00},//vert
  {0.00,0.00,1.00},//bleu
  {1.00,1.00,0.00},//jaune
  {0.00,1.00,1.00},//cyan
  {1.00,0.00,1.00},//magenta
};
```

## 3.4. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- On utilise 6 couleurs pour la base d'apprentissage. L'intervalle  $[0,255]$  doit être ramené à  $[0,1]$
- La valeur « 1 » en sortie signifie que la couleur est chaude sinon la sortie vaut « 0 »

Rouge : 255/0/0		<b>Chaude</b>	<pre>const float sortiesAttendues [6] [1] = {   {1},//rouge   {0},//vert   {0},//bleu   {1},//jaune   {0},//cyan   {1},//magenta };</pre>
Vert : 0/255/0		<b>Froide</b>	
Bleu : 0/0/255		<b>Froide</b>	
Jaune : 255/255/0		<b>Chaude</b>	
Cyan : 0/255/255		<b>Froide</b>	
Magenta : 255/0/255		<b>Chaude</b>	



## 3.4. Mis en œuvre de la bibliothèque NeuralNetwork

### ❑ Réseau 3-3-1

- Pour la première mise en œuvre de cette bibliothèque, nous commençons par utiliser une couche cachée de 3 neurones.
- On définit le réseau de neurones en plus de la base d'apprentissage

```
#include <NeuralNetwork.h>
#define ITER 4000      Nombre d'utilisation de la base d'apprentissage
unsigned int couches[] = {3, 3, 1};
float *sorties;
NeuralNetwork NN(couches, 3);
```

Nombre de neurones par couche

- On calcul la valeur des poids et bias

```
for (int i=0; i<ITER; i++) {
  for (int j = 0; j < 6; j++) {
    NN.FeedForward (entrees[j]);
    NN.BackProp (sortiesAttendues[j]);
  }
}
```

6 lignes dans la base d'apprentissage  
Calcul de la sortie  
Modification des poids et bias

### 3.4. Mis en œuvre de la bibliothèque NeuralNetwork

#### ❑ Réseau 3-3-1

- On calcule et imprime la valeur de la sortie pour la base d'apprentissage afin de vérifier si cela est conforme avec les valeurs attendues.

```
for (int i = 0; i<6; i++) {  
    sorties = NN.FeedForward(entrees[i]);    Calcul de la sortie pour la base  
    Serial.print(i);                          d'apprentissage  
    Serial.print(": ");  
    Serial.println(sorties[0],7); Impression de la première valeur du tableau  
    }                                         « sorties » avec 7 chiffres significatifs (Il n'y a  
                                           qu'une valeur dans le tableau car il n'y a  
                                           qu'un neurone en sortie
```

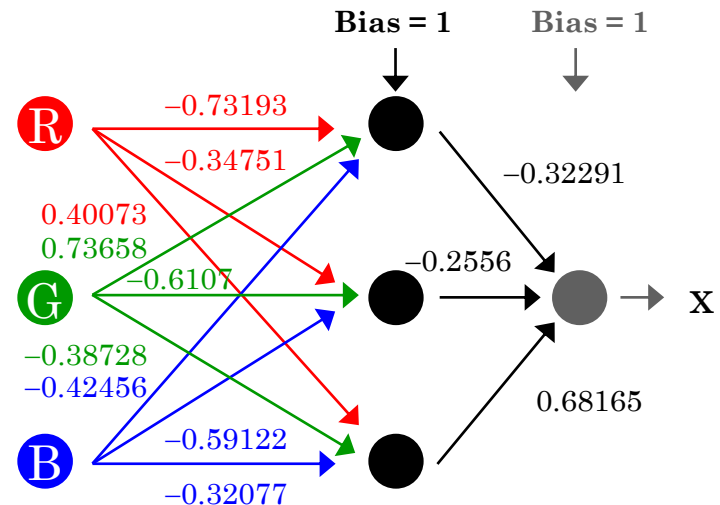
- Finalement, on demande l'impression des poids et bias :

```
NN.print();
```

## 3.4. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Réseau 3-3-1

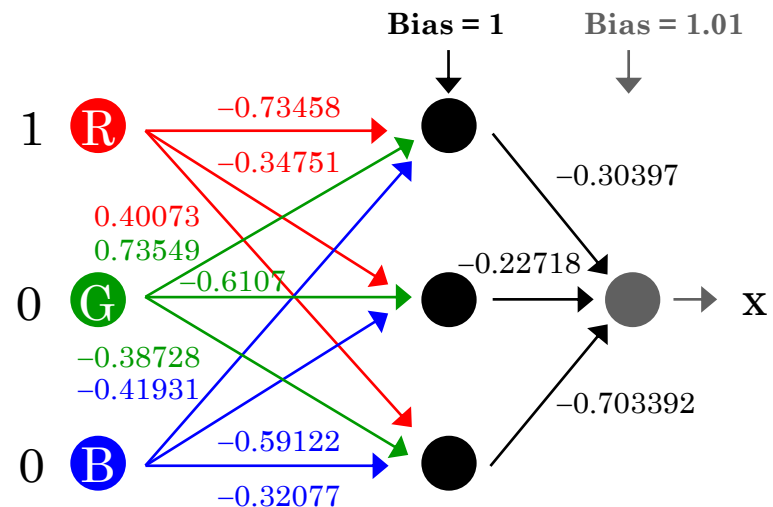
- A présent, nous regardons la modification des poids et bias pour l'utilisation des 3 premières lignes de la base d'apprentissage.
- La librairie initialise aléatoirement tous les poids et les bias du réseau .



## 3.4. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Réseau 3-3-1

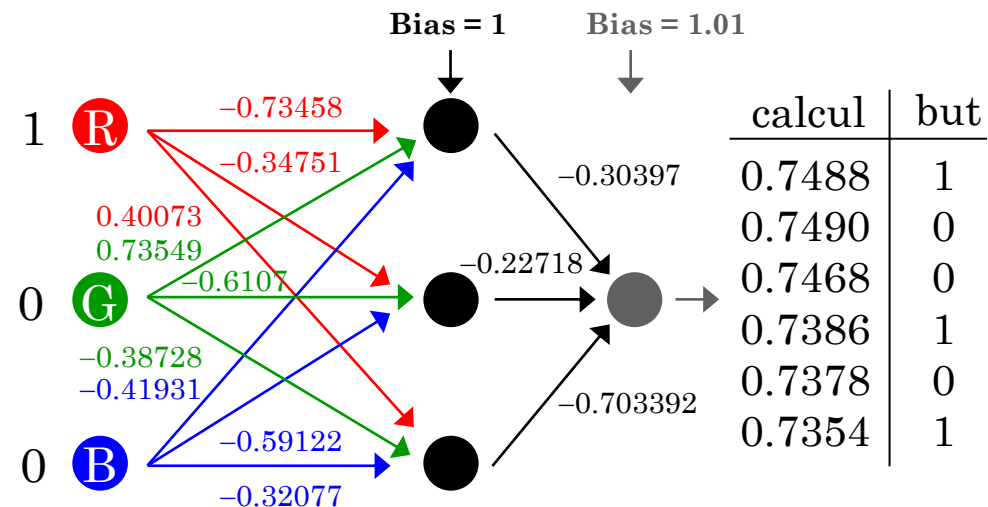
- A présent, nous regardons la modification des poids et bias pour l'utilisation des 3 premières lignes de la base d'apprentissage.
- La librairie initialise aléatoirement tous les poids et les bias du réseau .
- On effectue une première modification des poids et bias avec la couleur rouge.



## 3.4. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Réseau 3-3-1

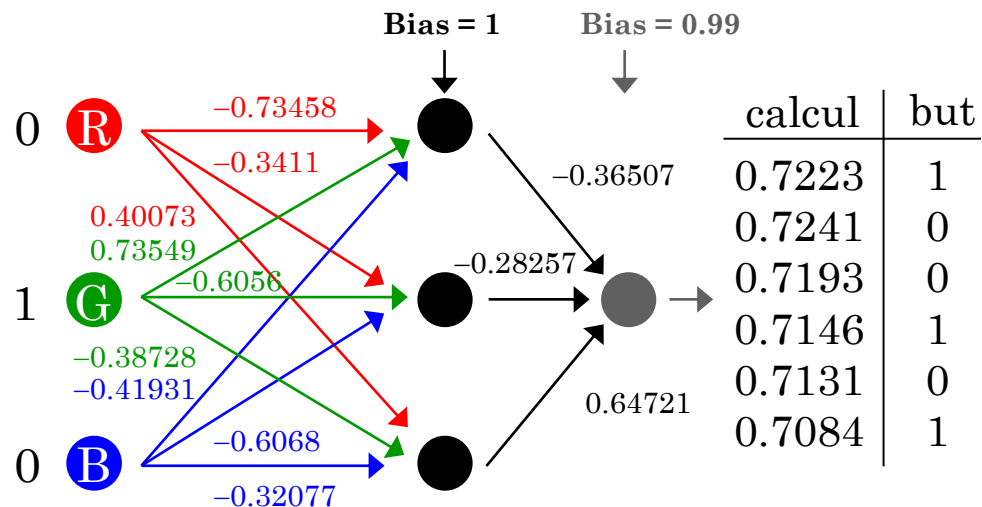
- A présent, nous regardons la modification des poids et bias pour l'utilisation des 3 premières lignes de la base d'apprentissage.
- La librairie initialise aléatoirement tous les poids et les bias du réseau .
- On effectue une première modification des poids et bias avec la couleur rouge.
- On regarde alors la valeur de la sortie sur toute la base d'apprentissage.



## 3.4. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Réseau 3-3-1

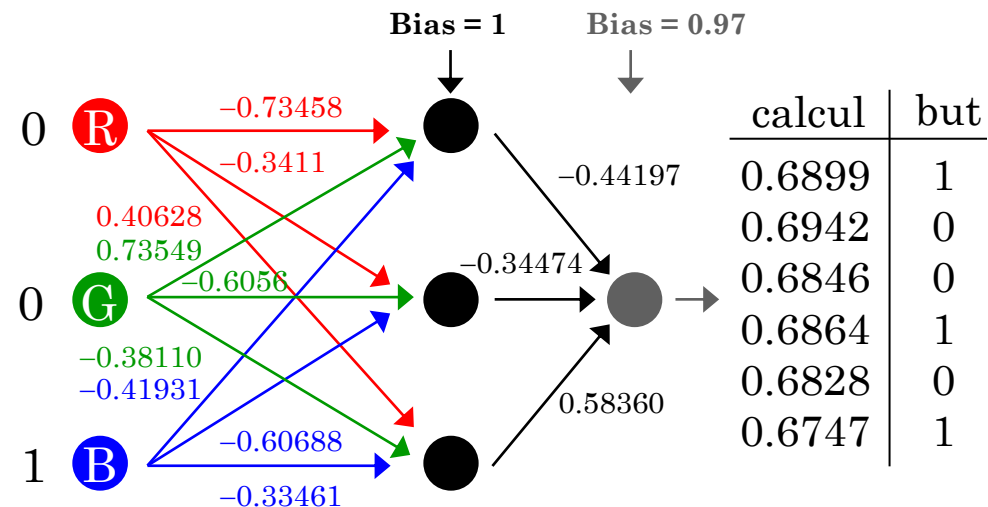
- A présent, nous regardons la modification des poids et bias pour l'utilisation des 3 premières lignes de la base d'apprentissage.
- La librairie initialise aléatoirement tous les poids et les bias du réseau .
- On effectue une première modification des poids et bias avec la couleur rouge.
- On regarde alors la valeur de la sortie sur toute la base d'apprentissage.
- On procède de même pour le vert.



## 3.4. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Réseau 3-3-1

- A présent, nous regardons la modification des poids et bias pour l'utilisation des 3 premières lignes de la base d'apprentissage.
- La librairie initialise aléatoirement tous les poids et les bias du réseau .
- On effectue une première modification des poids et bias avec la couleur rouge.
- On regarde alors la valeur de la sortie sur toute la base d'apprentissage.
- On procède de même pour le vert.
- Puis avec le bleu.
- En ainsi de suite sur toute la base.

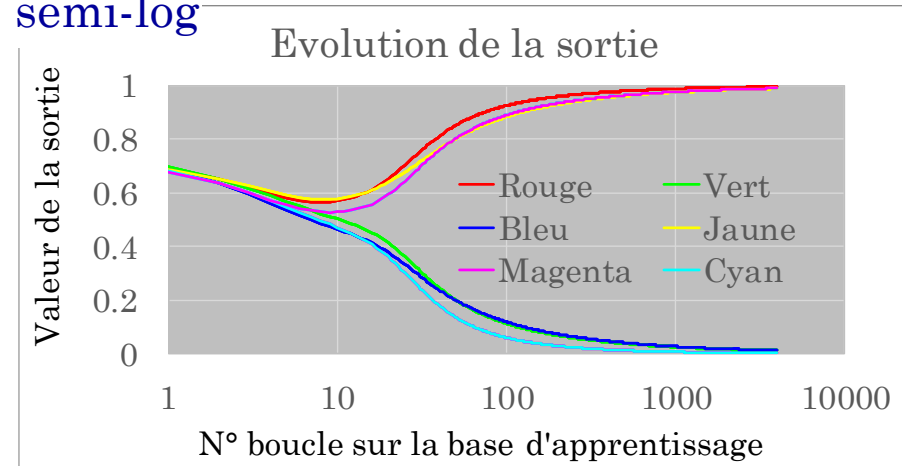


## 3.4. Mis en œuvre de la bibliothèque NeuralNetwork

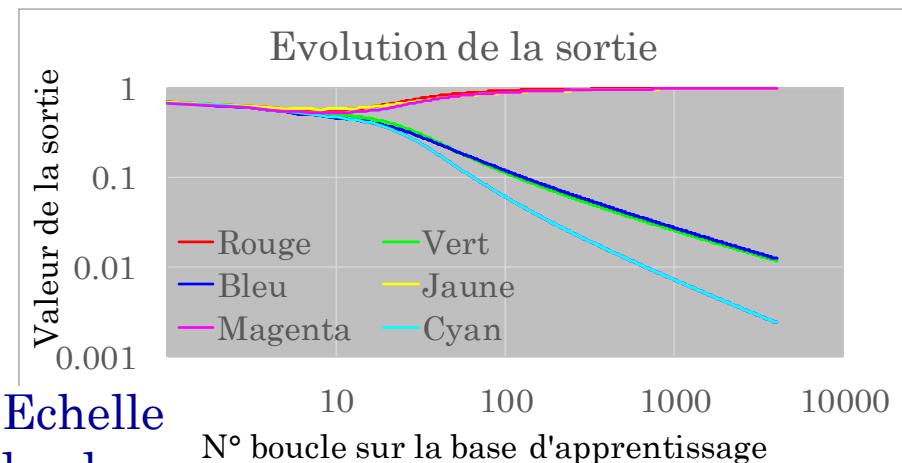
### □ Réseau 3-3-1

- Nous représentons maintenant l'évolution du calcul de la sortie en fonction de la couleur et en fonction du nombre d'utilisation de la base d'apprentissage.
- Nous remarquons que les sorties convergent vers les valeurs 1 et 0 si les couleurs sont chaudes ou froides
- Le calcul des 14 poids et bias a nécessité 45.1 s pour 4000 utilisations de la base d'apprentissage

Echelle  
semi-log



Echelle  
log-log





## 3.4. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Réseau 3-3-1

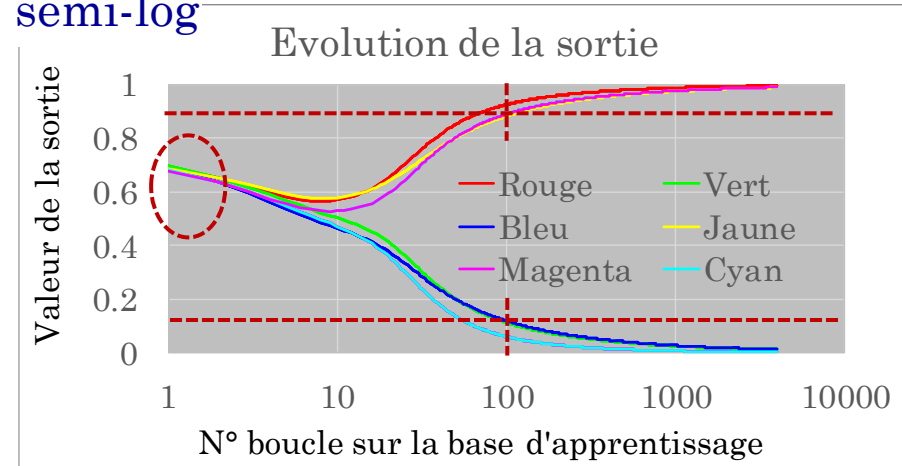
■ L'arrêt de la modification des poids et bias peut se faire :

- ✓ Soit en fixant un nombre total de boucles.
- ✓ Soit en fixant une erreur maximale à ne pas dépasser entre les sorties calculées et celles souhaitées.

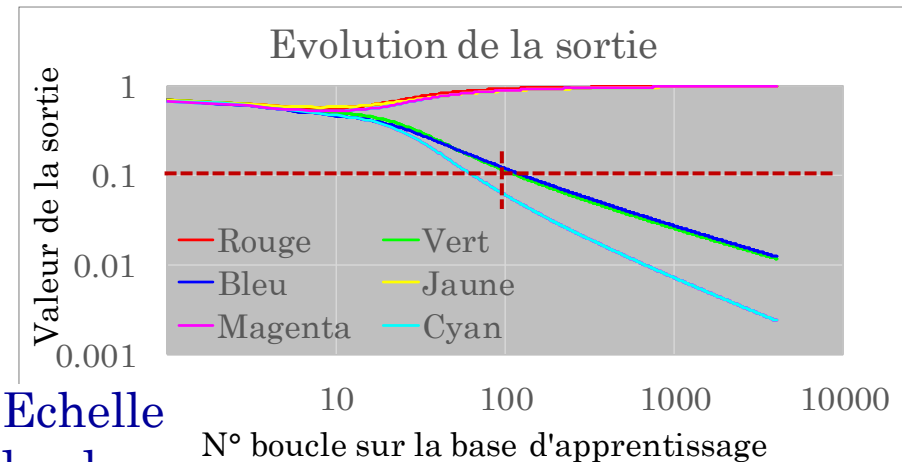
■ On applique par la suite un seuil égal à 0.5 sur la sortie pour savoir si la couleur est chaude ou froide :

- ✓  $\text{Sortie} \geq 0.5$  : couleur chaude
- ✓  $\text{Sortie} < 0.5$  : couleur froide

Echelle  
semi-log



Echelle  
log-log

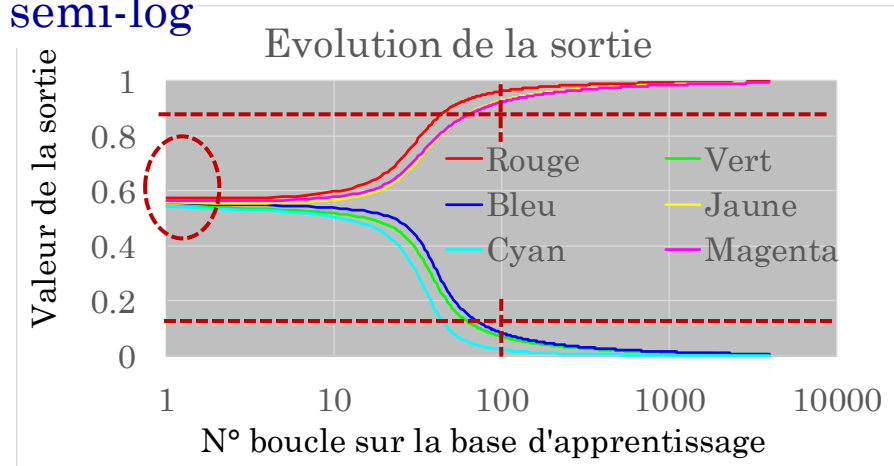


## 3.4. Mis en œuvre de la bibliothèque NeuralNetwork

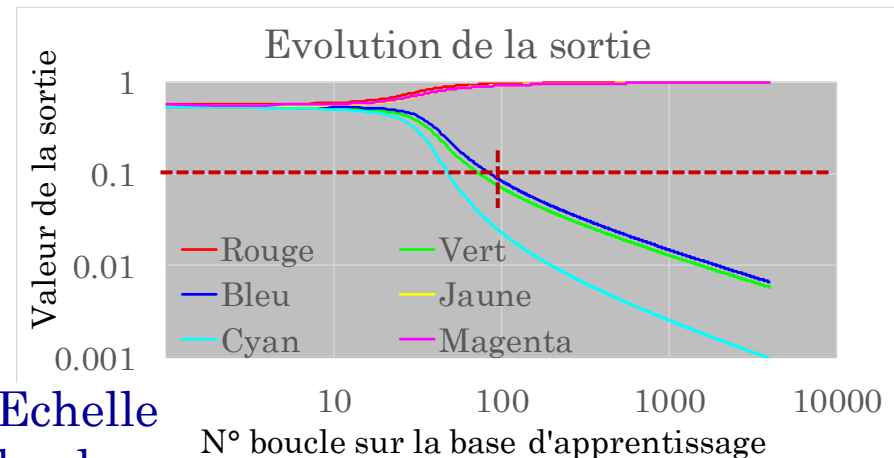
### ❑ Réseau 3-12-12-1

- On peut se demander ce qui se passe si on augmente le nombre de couches et de neurones cachés.
- Ce changement n'induit pas une amélioration importante de la précision sur la sortie.
- La rapidité de convergence semble plutôt due aux valeurs des poids choisis par la librairie à l'initialisation
- Par contre, le calcul des 195 poids et bias a nécessité 7mn et 47s pour 4000 utilisations de la base d'apprentissage

Echelle  
semi-log



Echelle  
log-log



## 3.5. Alors ? Couleurs chaudes ou froides

### □ Mesure de la couleur et utilisation du réseau de neurones

- L'utilisation des poids et bias calculés dans la « loop » nécessite :
  - ✓ Soit de recalculer ces valeurs à chaque redémarrage de la carte arduino.
  - ✓ Soit de redéfinir un réseau de neurone et de donner la valeur des poids et bias dans un autre programme que celui du calcul du réseau.

Mesure et mise en forme des  
valeurs RGB

Définition du tableau des  
valeurs RGB

Calcul de la sortie avec le réseau  
Utilisation du seuil à 0.5

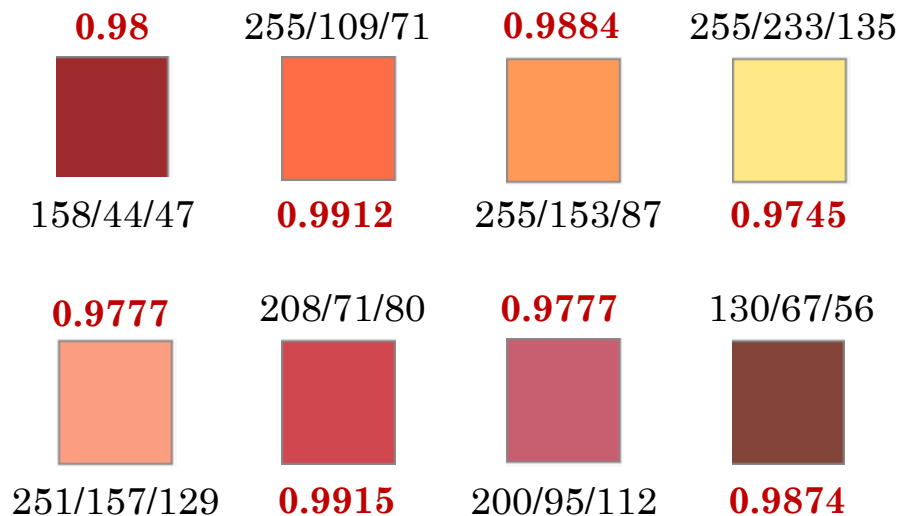
```
tcs.getRawData(&r,&g,&b,&c);
int rmap=map(r,rmin,rmax,0,255);
int gmap=map(g,gmin,gmax,0,255);
int bmap=map(b,bmin,bmax,0,255);
entreesmesurees[0]=constrain(rmap,0,255)/255.0;
entreesmesurees[1]=constrain(gmap,0,255)/255.0;
entreesmesurees[2]=constrain(bmap,0,255)/255.0;
sorties = NN.FeedForward(entreesmesurees);
Serial.println(sorties[0]>0.5?"Chaude" : "Froide");
delay(1000);
```

## 3.5. Alors ? Couleurs chaudes ou froides ?

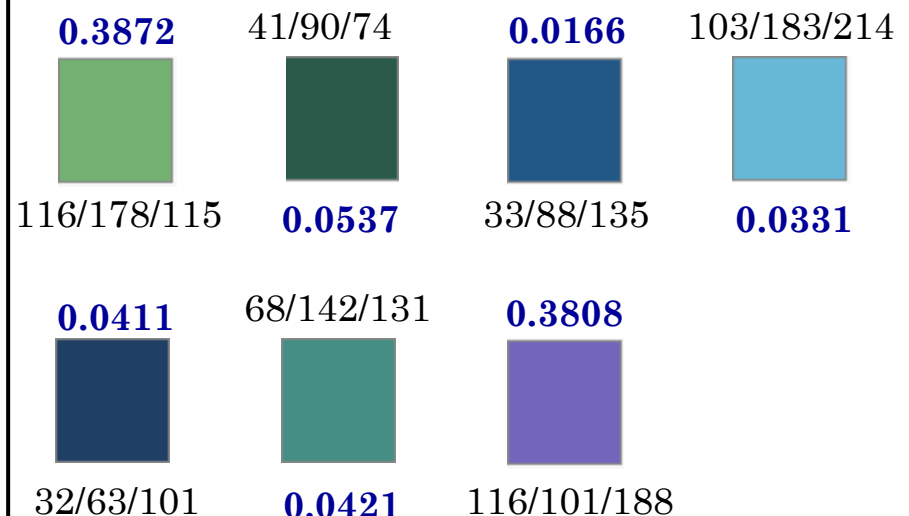
### □ Réseau 3-3-1

- Il nous reste à vérifier si le réseau de neurones détecte bien si une couleur est chaude ou froide avec la base de couleurs introduites dans ce chapitre.
- Nous indiquons ici en gras les valeurs calculées pour la sortie.
- Le résultat est conforme aux attentes.

#### Couleurs chaudes



#### Couleurs froides

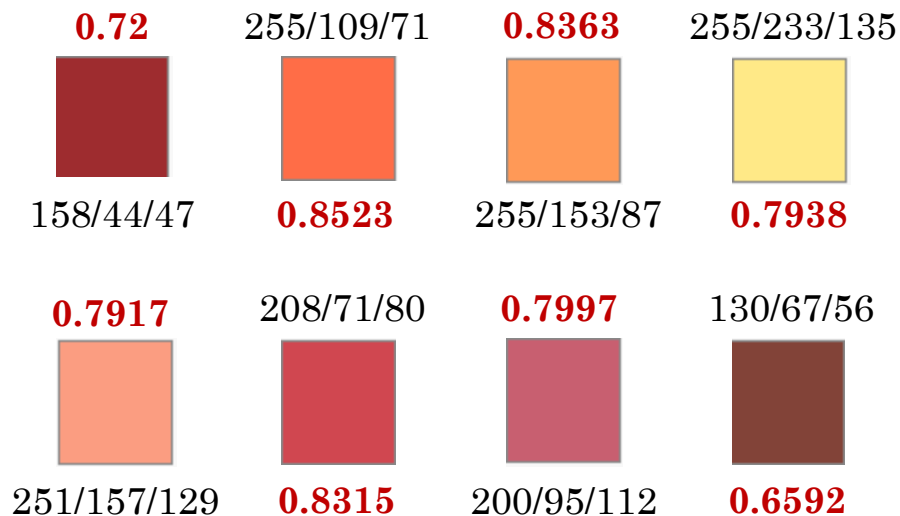


## 3.5. Alors ? Couleurs chaudes ou froides ?

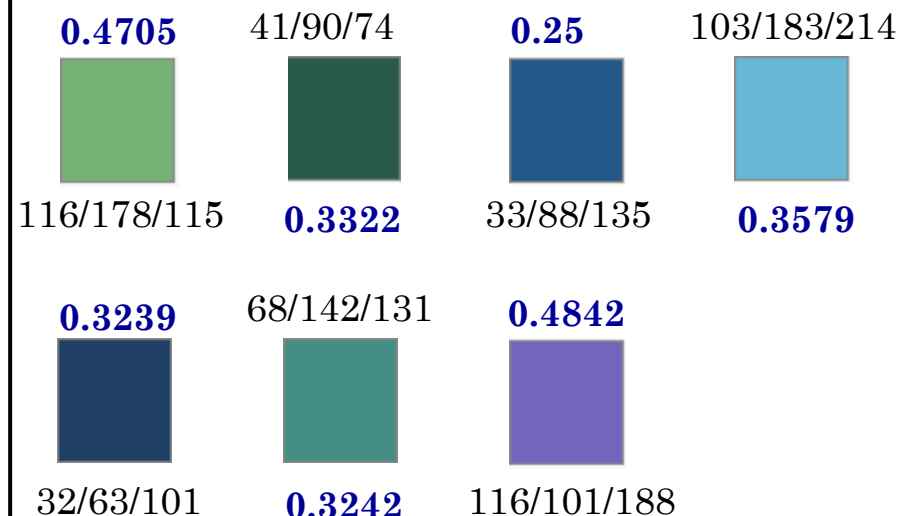
### □ Réseau 3-12-12-1

- L'augmentation du nombre de couches et de neurones par sous-couche ne change rien pour cette application.
- Nous indiquons ici en gras les valeurs calculées pour la sortie.
- On notera qu'elles sont plus proches de 0.5 que pour le réseau 3-3-1.

### Couleurs chaudes

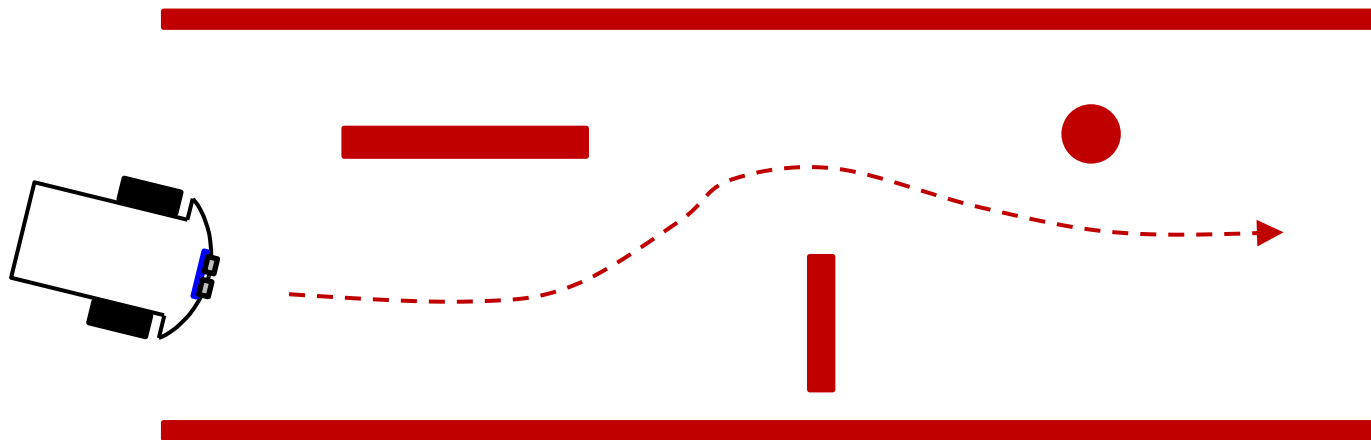


### Couleurs froides



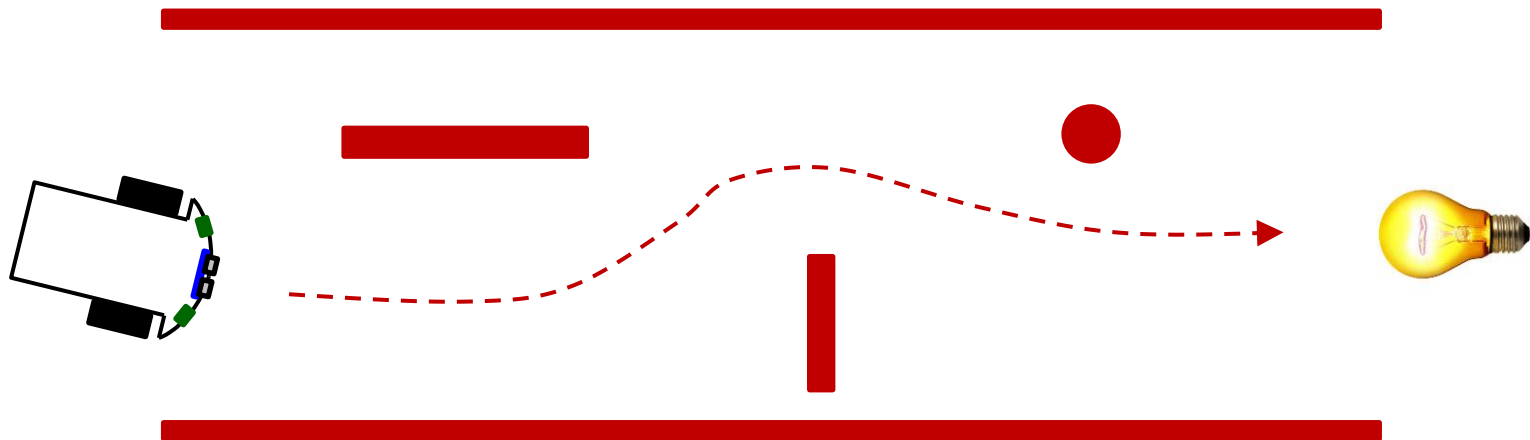
## 4.1. Introduction

- Le but est de faire avancer la voiture du cours en évitant des obstacles.
- La voiture est équipée d'un capteur de distance monté sur un servomoteur 9g (c.f. cours éléments de robotique avec arduino : Moteurs).
- Le capteur est donc orientable ce qui évite l'utilisation de plusieurs capteurs.



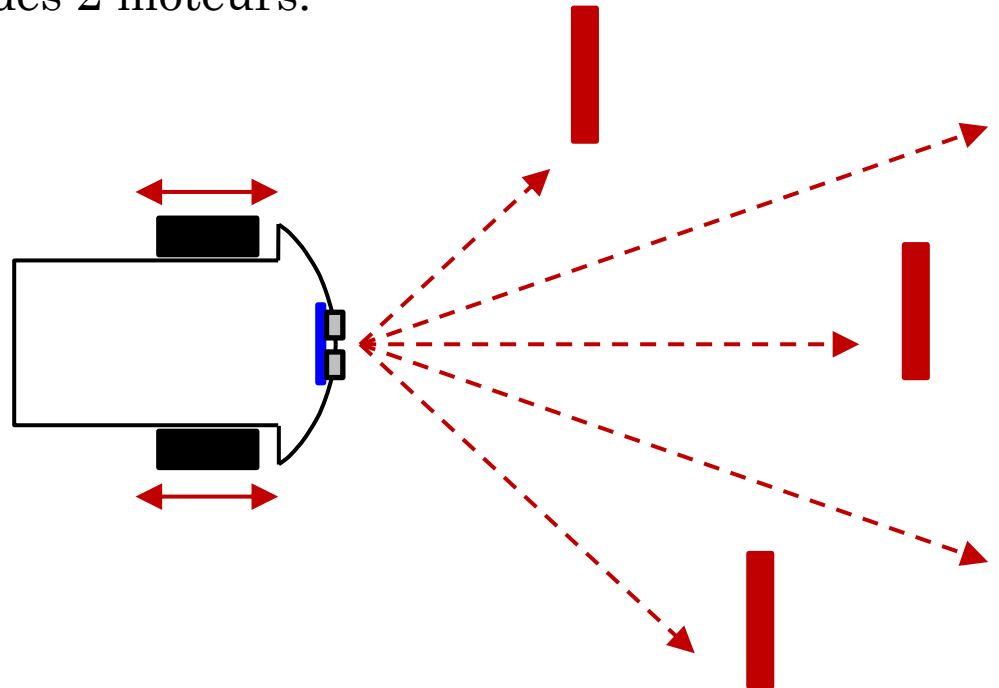
## 4.1. Introduction

- Le but est de faire avancer la voiture du cours en évitant des obstacles.
- La voiture est équipée d'un capteur de distance monté sur un servomoteur 9g (c.f. cours éléments de robotique avec arduino : Moteurs).
- Le capteur est donc orientable ce qui évite l'utilisation de plusieurs capteurs.
- On ne donnera pas de point d'arrivée à la voiture mais on peut parfaitement lui demander de se diriger vers de la lumière (montage avec 2 LRD).



## 4.1. Introduction

- Le réseau de neurones va permettre de construire une équation complexe qui indiquera aux 2 moteurs à quelle vitesse ils doivent tourner et dans quel sens en fonction de la distance des obstacles.
- La base d'apprentissage doit définir des distances clés (en fonction de l'angle) et les valeurs vitesse/sens des 2 moteurs.
- En fonction du comportement de la voiture, cette base pourra être modifiée ou complétée.
- La pièce maîtresse du montage est le capteur de distance qui peut être basé soit sur une onde acoustique (ultrason) ou lumineuse (laser IR).

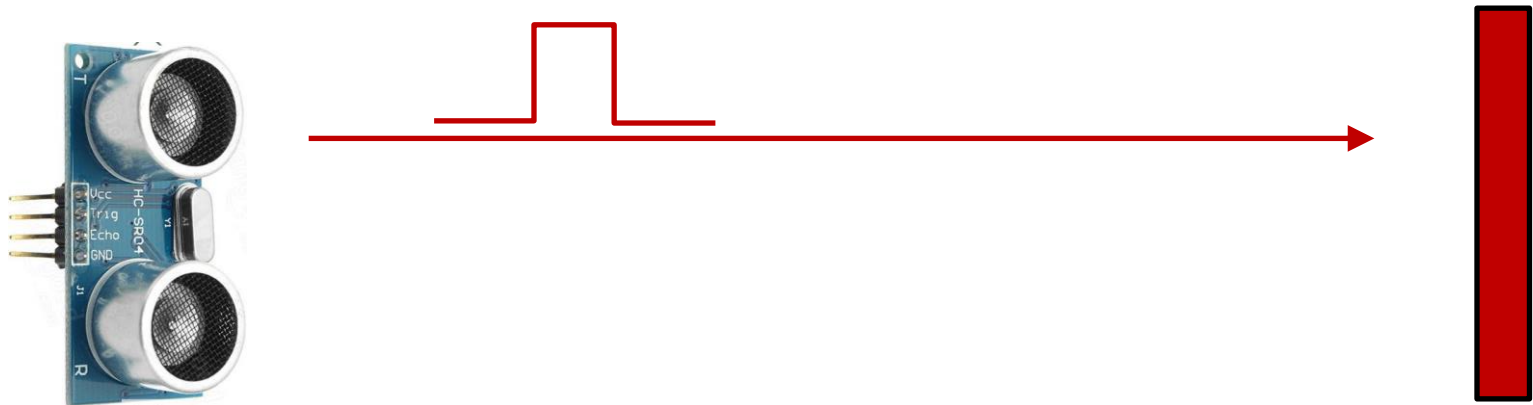




## 4.2. Montage et capteur

### □ Capteur HC-SR04

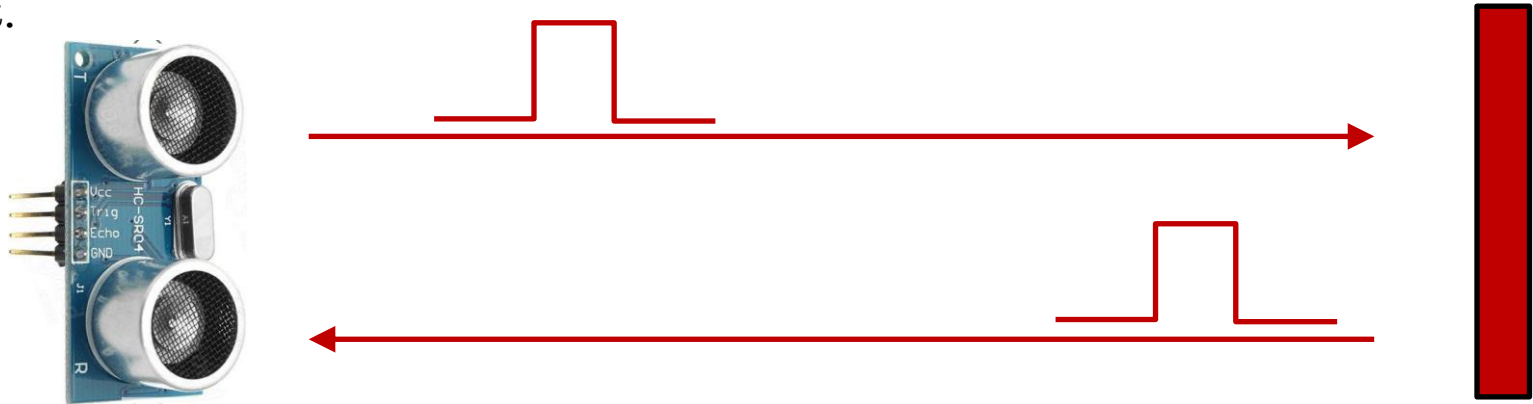
- Il faudra se référer au cours « électronique avec arduino » pour une description plus détaillée du capteur et de son utilisation.
- Une onde acoustique est envoyée par un haut-parleur piézo-électrique.



## 4.2. Montage et capteur

### □ Capteur HC-SR04

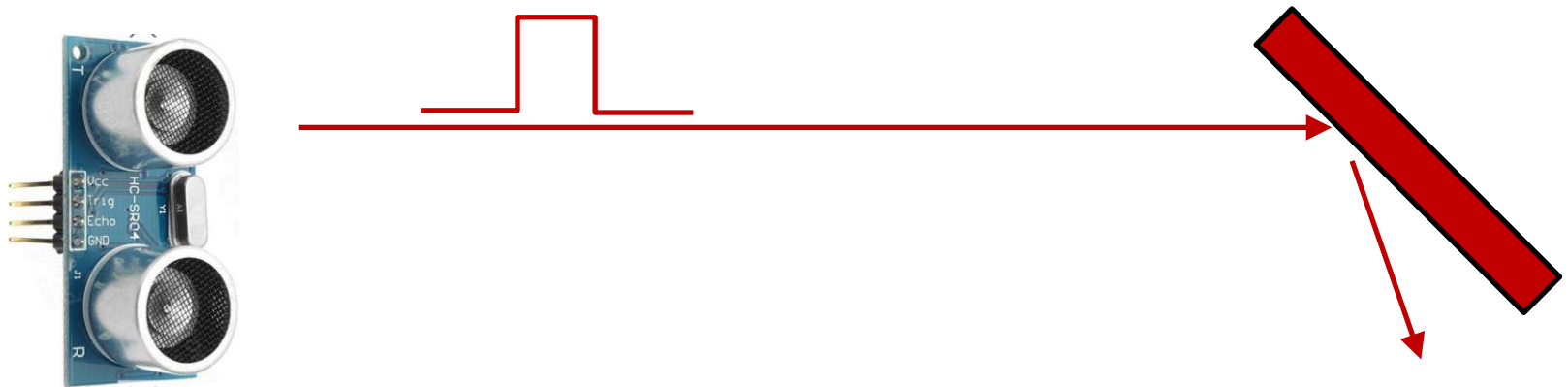
- Il faudra se référer au cours « électronique avec arduino » pour une description plus détaillée du capteur et de son utilisation
- Une onde acoustique est envoyée par un haut-parleur piézo-électrique.
- Elle est réfléchiée sur l'obstacle revient sur un autre haut-parleur piézo-électrique.
- En prenant en compte la vitesse de propagation dans l'air (environ 340 m/s), le temps que met l'onde pour faire l'aller-retour donne la distance de l'objet.



## 4.2. Montage et capteur

### □ Capteur HC-SR04

- L'orientation de l'obstacle (ou sa composition) peut conduire à une mauvaise évaluation de la distance.
- Dans ce cas, la distance renvoyée par la librairie (NewPing) sera égale à 0
- Il est alors préférable de refaire la mesure (au maximum 10 fois par exemple) jusqu'à obtenir une valeur non nulle ou utiliser la mesure précédente
- Cela peut évidemment prendre du temps sans garantie de succès. La mesure risque au finale d'être aberrante ainsi que le comportement de la voiture ...

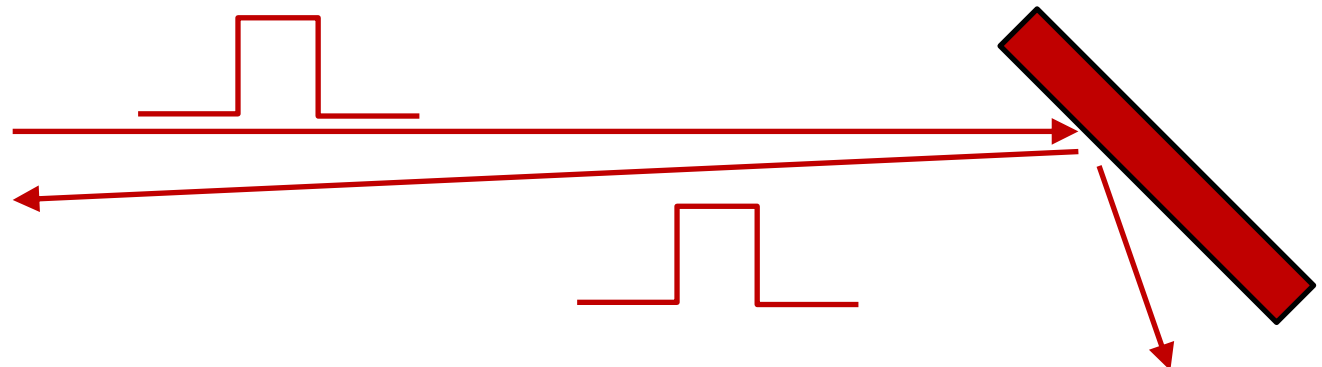
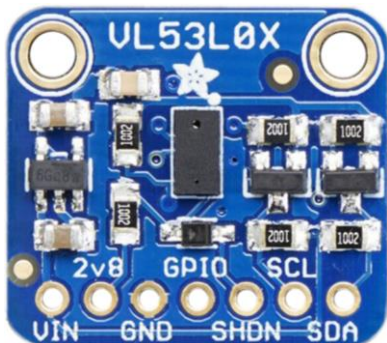


## 4.2. Montage et capteur

### ❑ Capteur VL53L0X



- Une autre option est d'utiliser un capteur basé sur un laser IR
- La matière de l'obstacle ainsi que son orientation a aussi une influence sur la mesure de distance mais la mesure est bien plus fiable qu'avec le capteur à ultra-son sauf si l'obstacle est en verre (cas des fenêtres !)
- Pour éviter cet autre problème, on peut utiliser les distances des 2 types de capteurs et de vérifier la cohérence pour savoir quelle valeur utiliser (fusion de capteurs). Nous ferons au plus simple dans la suite en utilisant uniquement le capteur laser.

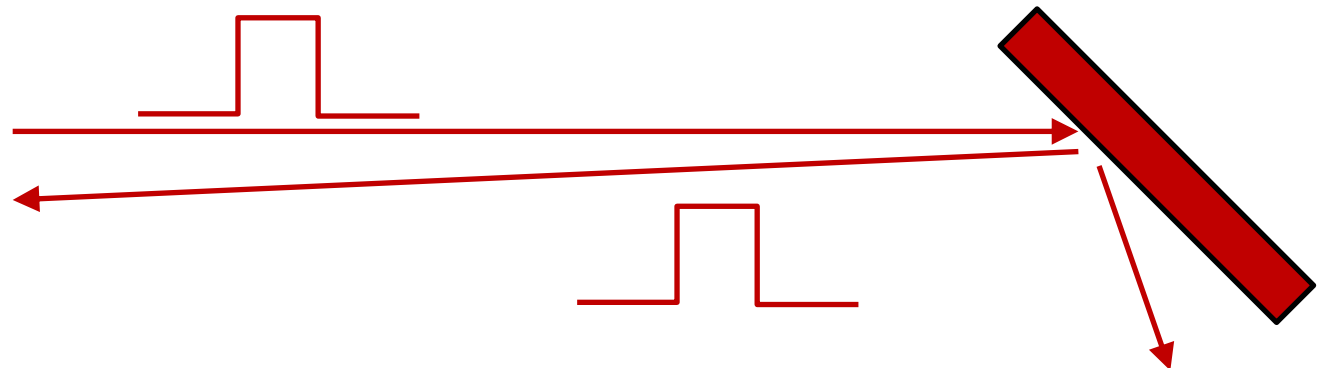
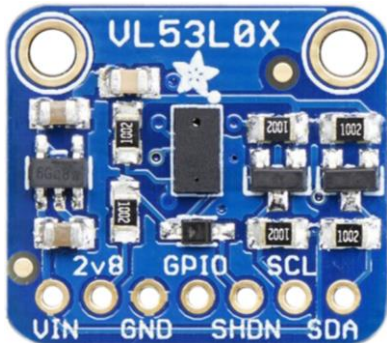


## 4.2. Montage et capteur

### □ Capteur VL53L0X



- Les principales caractéristiques du modules sont :
  - ✓ Distance maximale de mesure : 2 m (support blanc en intérieur)
  - ✓ Communication par bus I<sup>2</sup>C (bibliothèque wire.h) avec l'adresse 0x29
  - ✓ Laser IR, à 940 nm, de classe 1
  - ✓ Le temps max d'une mesure est de 33 ms (valeur = 8190 mm si non mesurable)
- Nous utiliserons la bibliothèque VL53L0X.h [github.com/pololu/vl53l0x-arduino](https://github.com/pololu/vl53l0x-arduino)



## 4.2. Montage et capteur

### ❑ Fonctions de la bibliothèque VL53L0X.h

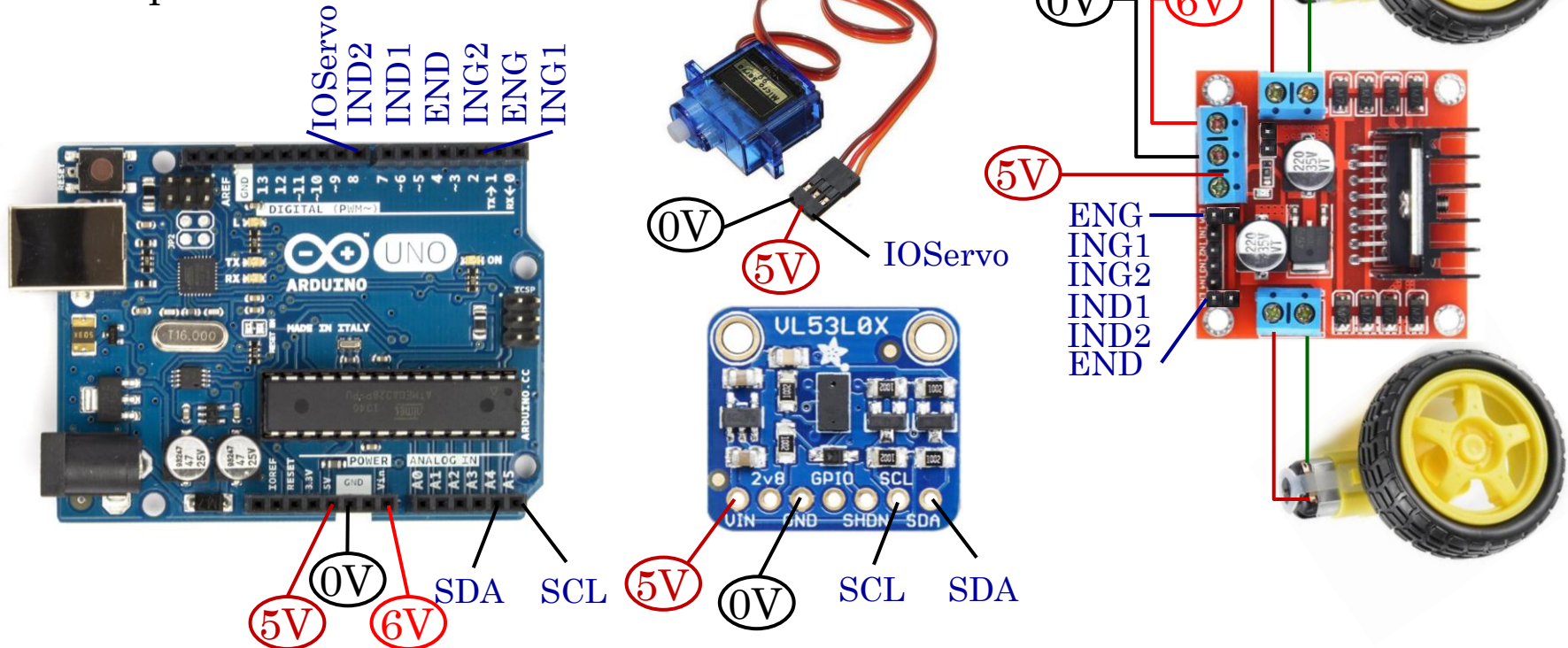
- **VL53L0X** X; permet de déclarer le capteur en lui donnant un nom (X)
- **X.setTimeout**(Y) qui s'applique au capteur de nom X et permet de définir le temps maximum passé à faire la mesure. Y doit être indiqué en ms (500 ms par exemple)
- **X.startContinuous**(Z) qui s'applique au capteur de nom X et permet de définir l'intervalle de temps entre 2 mesures, Z en ms. Si (Z = 0) ou () ,alors le capteur fait des mesures en permanence.
- **X.stopContinuous**() arrête les mesures en continu.
- **X.readRangeContinuousMillimeters**() donne la valeur de la distance en mm dans le mode continu.
- **X.readRangeSingleMillimeters**() donne la valeur de la distance en mm lorsque qu'on est en mode mesure à la demande



## 4.2. Montage et capteur

### □ Montage

- Les 3 jumpers du module L298 (c.f. cours **éléments de robotique avec arduino : Moteurs**) sont enlevés. L'alimentation 5V de sa partie logique est fournie par la carte arduino.



## 4.2. Montage et capteur

### □ Servomoteur et mesure de distance

- Le module VL53L0X est monté sur le servomoteur qui doit avoir un angle de 90° pour être dans le prolongement de la voiture
- Dans ce chapitre, le servomoteur permettra la mesure de distances suivant 3 angles : 150° (à gauche), 90° (tout droit) et 30° (à droite)
- On déclare les librairies, les variables et constantes ainsi que le capteur de distance et le servomoteur :

```
#include <Servo.h>
#include <wire.h>
#include <VL53L0X.h>
Servo myservo;
VL53L0X sensor;
float cmG=200.0,cm=200.0,cmD=200.0;
cont int angleG=150,angle=90,angleD=30
cons float Dmax = 200.0;
```

Distance maximale de mesure



## 4.2. Montage et capteur

### □ Servomoteur et mesure de distance

- Le setup positionne le servomoteur à 90° :

```
void setup() {  
  Serial.begin(9600);  
  myservo.attach(10);  
  myservo.write(90);  
  Wire.begin();  
  sensor.setTimeout(500);  
  delay(500); }
```

- On définit une fonction qui prend comme paramètre l'angle de mesure et retourne la distance mesurée. Si la mesure est impossible (orientation/matière de l'obstacle ou distance supérieure à 2 m), la valeur retournée est 200 cm :

```
float measure(int Ang){  
  myservo.write(Ang);  
  delay(300);  
  float cmbis = sensor.readRangeSingleMillimeters()/10.0;  
  if (cmbis==819.0){cmbis=200.0;}  
  return cmbis; }
```

On laisse le temps au servomoteur d'atteindre l'angle

- On utilise cette fonction ainsi :      cmG=measure(angleG);

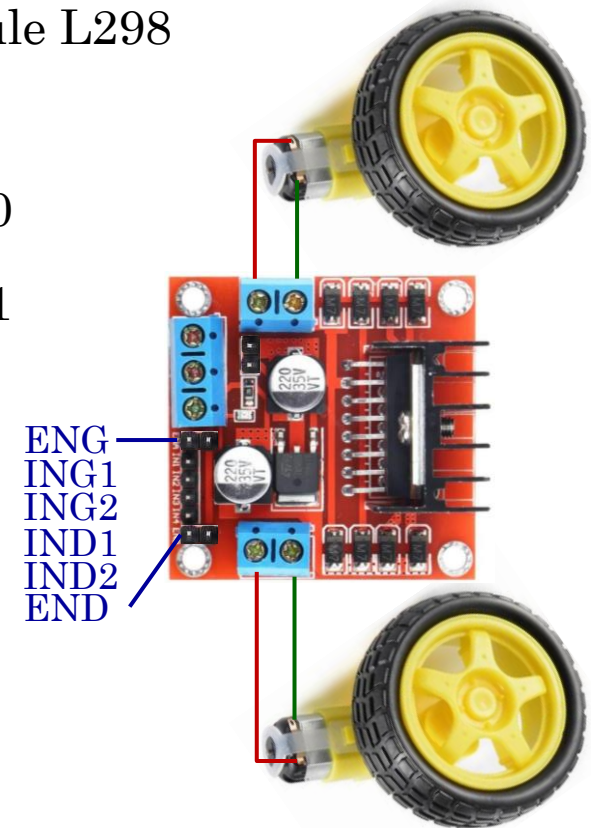
## 4.2. Montage et capteur

### □ Contrôle des moteurs

- Les sorties du réseau de neurones sont comprises entre 0 et 1 et il nous faut en tenir compte pour commander les entrées du module L298
- Cet intervalle est divisé en 2 pour les 2 sens :
  - ✓  $[0; 0.5[$  :  $ING1 = 0, ING2 = 1, IND1 = 1, IND2 = 0$
  - ✓  $[0.5; 1]$  :  $ING1 = 1, ING2 = 0, IND1 = 0, IND2 = 1$
- On règle alors la vitesse des moteurs en marche arrière avec (moteur G par exemple) :
 

```
analogWrite(ENG,mapf(SortieG,0,0.5,255,0))
```
- Et pour la marche avant (moteur D par exemple) :
 

```
analogWrite(END,mapf(SortieD,0.5,1,0,255))
```



## 4.2. Montage et capteur

### □ Contrôle des moteurs

- Pour simplifier l'écriture du programme, on peut utiliser cette fonction :

```
void moteur(float S_G,float S_D) {  
  if (S_G <0.5){  
    digitalWrite(ING1,HIGH);  
    digitalWrite(ING2,LOW);  
    analogWrite(ENG,mapf(S_G,0,0.5 ,255 ,0 ));}  
  else {  
    digitalWrite(ING1,LOW);  
    digitalWrite(ING2,HIGH);  
    analogWrite(ENG,mapf(S_G,0.5,1 ,0 ,255 ));}  
  if (S_D <0.5){  
    digitalWrite(IND1,LOW);  
    digitalWrite(IND2,HIGH);  
    analogWrite(END,mapf(S_D,0,0.5 ,255 ,0 ));}  
  else {  
    digitalWrite(IND1,HIGH);  
    digitalWrite(IND2,LOW);  
    analogWrite(END,mapf(S_D,0.5,1 ,0 ,255 ));} } }
```

## 4.2. Montage et capteur

### □ Contrôle des moteurs

- La fonction (à intégrer dans le programme) `mapf` permet d'utiliser des nombre à virgule pour les sorties du réseau de neurone:

```
int mapf(double val, double in_min, double in_max, double out_min, double out_max) {  
  
    return (val - in_min) * (out_max - out_min) / (in_max - in_min) + out_min; }
```

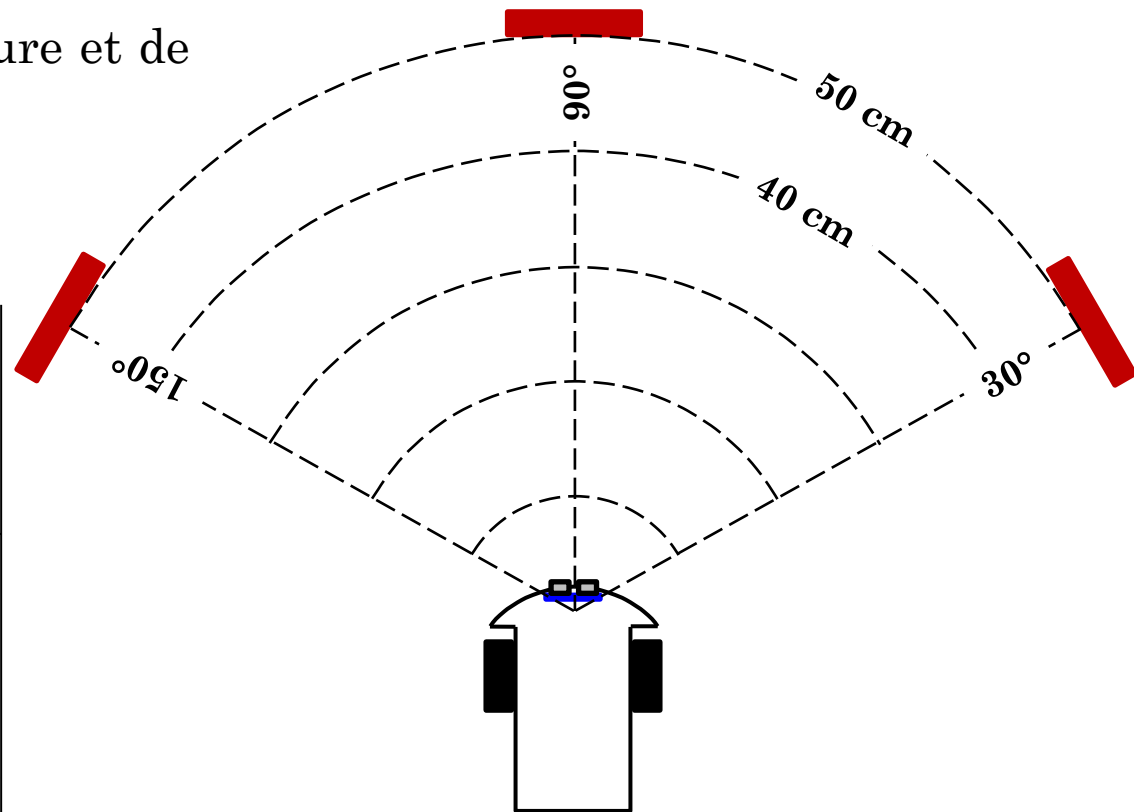
- Pour rappel, la fonction « `map` » n'utilise et ne renvoie que des entiers

## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- Elle correspond au fonctionnement que l'on souhaite avoir pour la voiture en fonction de l'angle de mesure et de la distance des obstacles
- On définit plusieurs actions

150°	90°	30°	G	D
50	50	50		

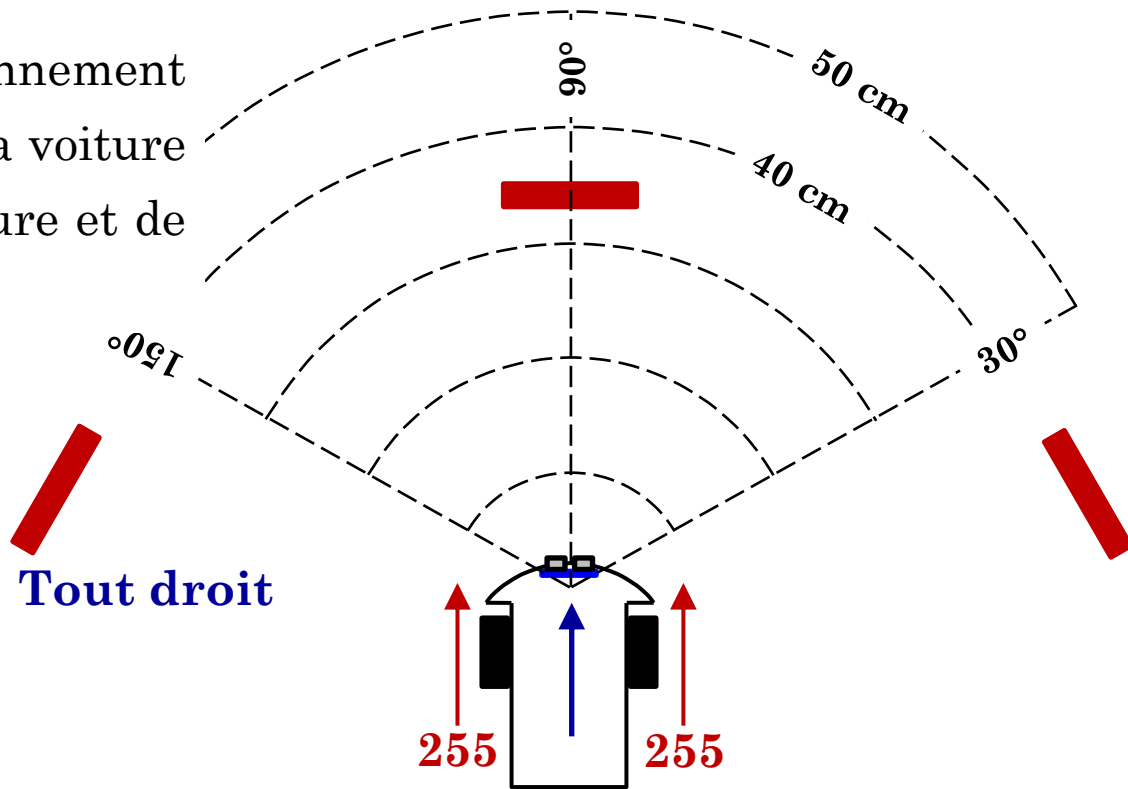


## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- Elle correspond au fonctionnement que l'on souhaite avoir pour la voiture en fonction de l'angle de mesure et de la distance des obstacles
- On définit plusieurs actions

150°	90°	30°	G	D
50	50	50	1	1



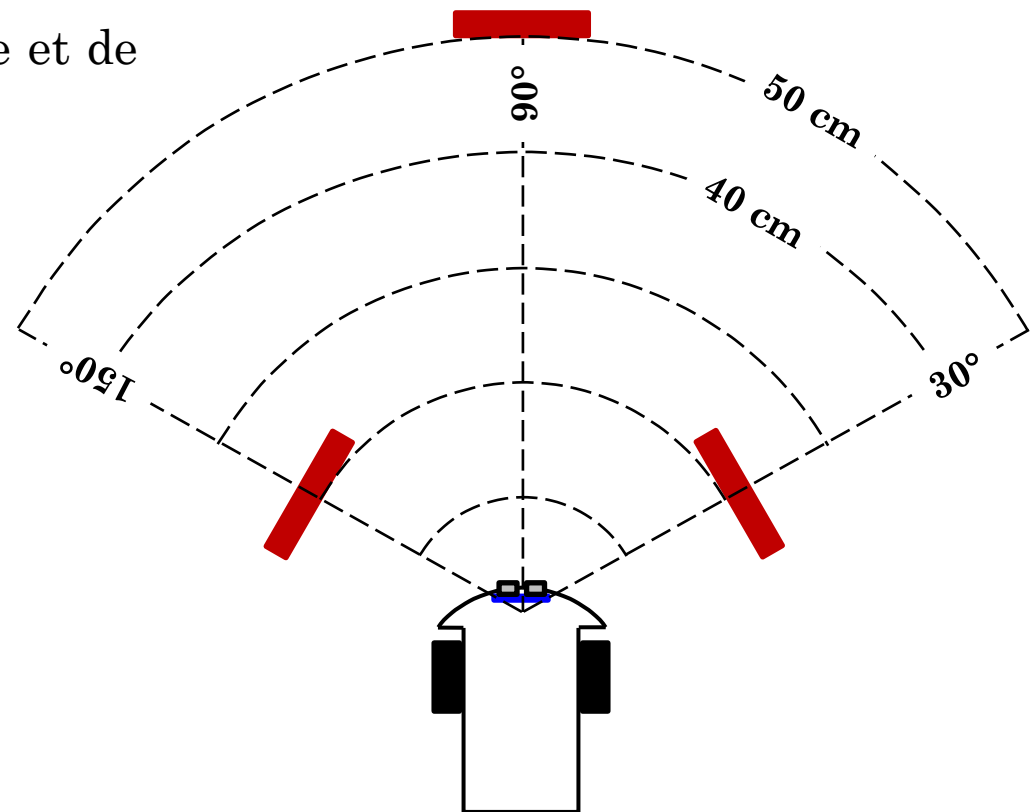
**Tout droit**

## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- Elle correspond au fonctionnement que l'on souhaite avoir pour la voiture en fonction de l'angle de mesure et de la distance des obstacles
- On définit plusieurs actions

150°	90°	30°	G	D
50	50	50	1	1
20	50	20		



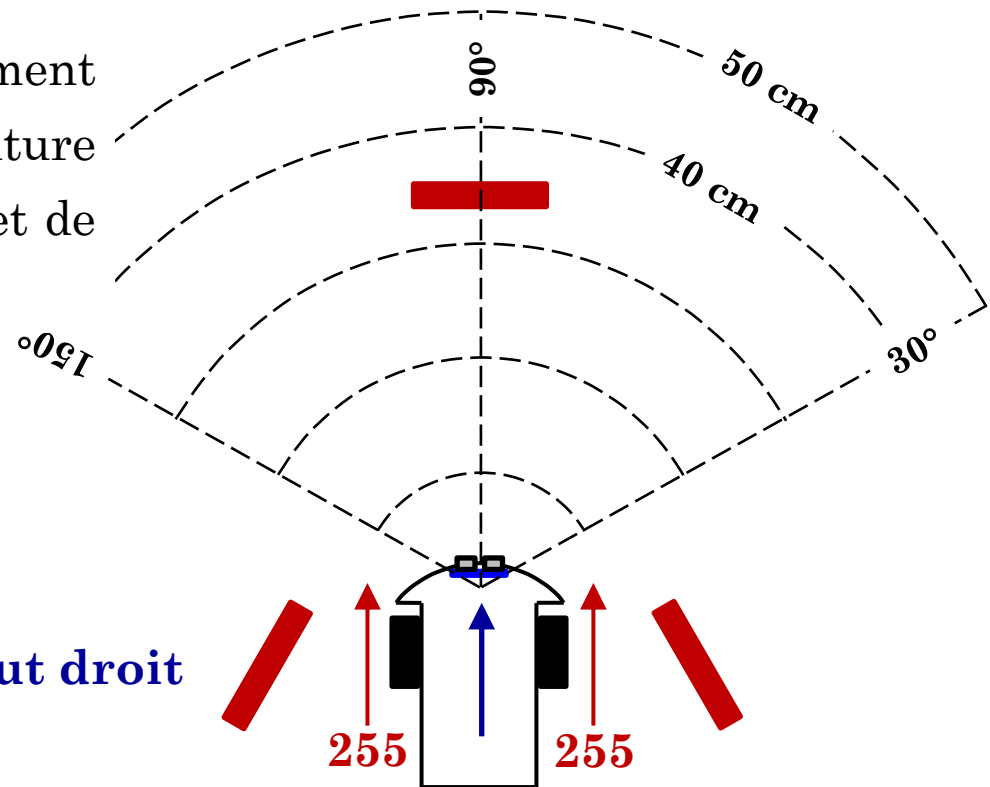
## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- Elle correspond au fonctionnement que l'on souhaite avoir pour la voiture en fonction de l'angle de mesure et de la distance des obstacles
- On définit plusieurs actions

150°	90°	30°	G	D
50	50	50	1	1
20	50	20	1	1

Tout droit



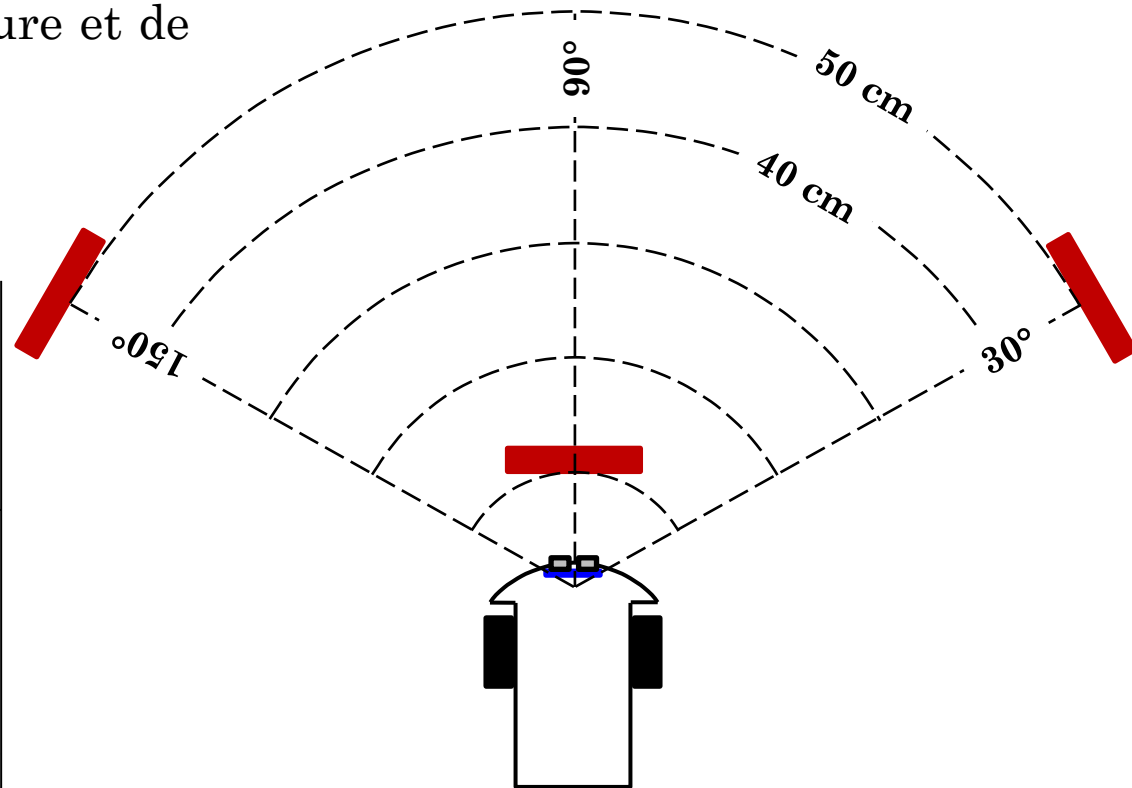


## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- Elle correspond au fonctionnement que l'on souhaite avoir pour la voiture en fonction de l'angle de mesure et de la distance des obstacles
- On définit plusieurs actions

150°	90°	30°	G	D
50	50	50	1	1
20	50	20	1	1
50	10	50		

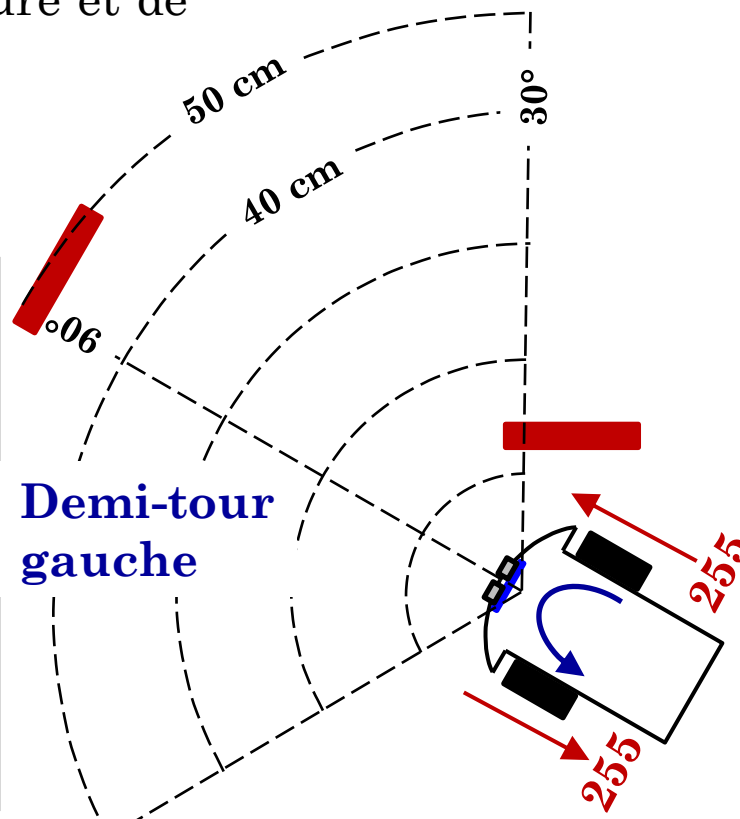


## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- Elle correspond au fonctionnement que l'on souhaite avoir pour la voiture en fonction de l'angle de mesure et de la distance des obstacles
- On définit plusieurs actions

150°	90°	30°	G	D
50	50	50	1	1
20	50	20	1	1
50	10	50	0	1

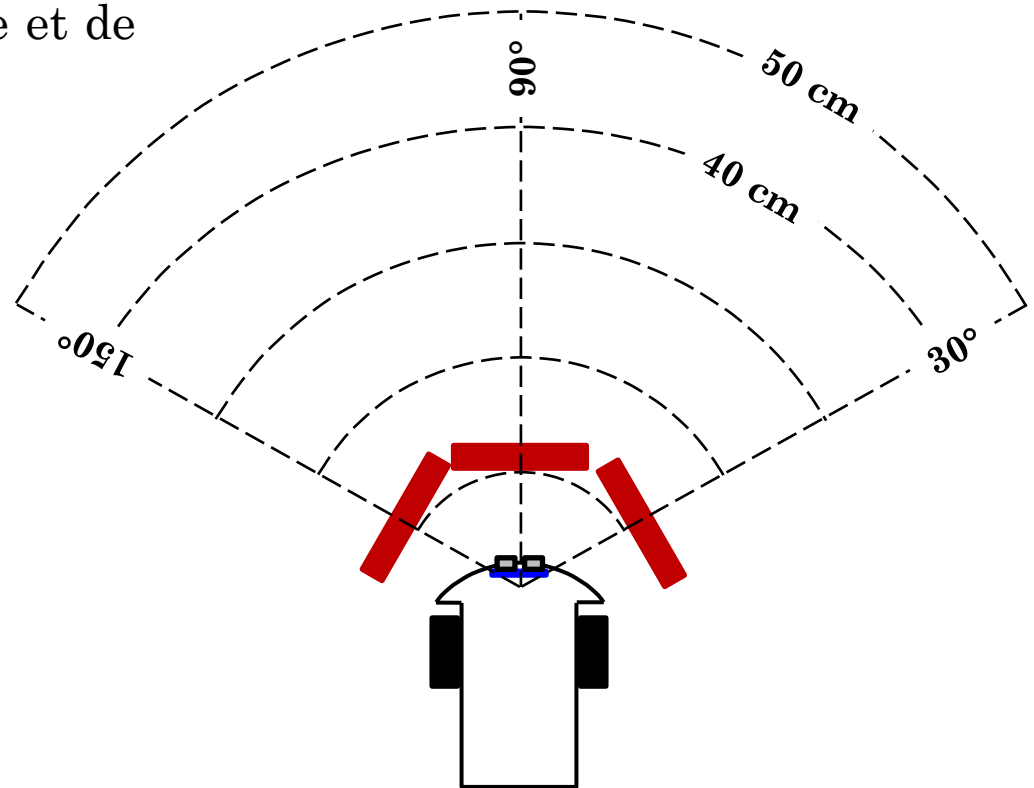


## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- Elle correspond au fonctionnement que l'on souhaite avoir pour la voiture en fonction de l'angle de mesure et de la distance des obstacles
- On définit plusieurs actions

150°	90°	30°	G	D
50	50	50	1	1
20	50	20	1	1
50	10	50	0	1
10	10	10		

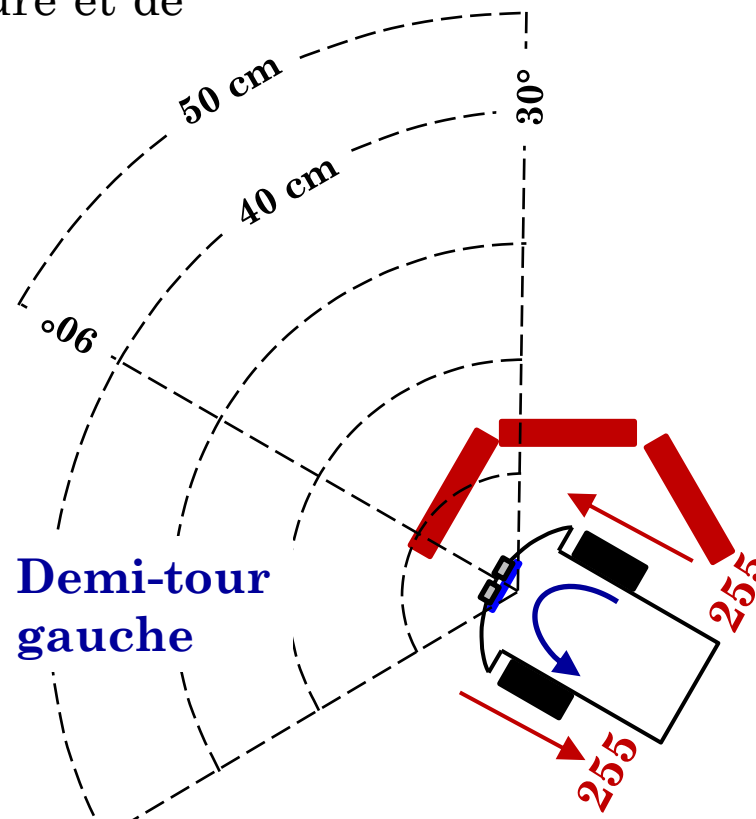


## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- Elle correspond au fonctionnement que l'on souhaite avoir pour la voiture en fonction de l'angle de mesure et de la distance des obstacles
- On définit plusieurs actions

150°	90°	30°	G	D
50	50	50	1	1
20	50	20	1	1
50	10	50	0	1
10	10	10	0	1

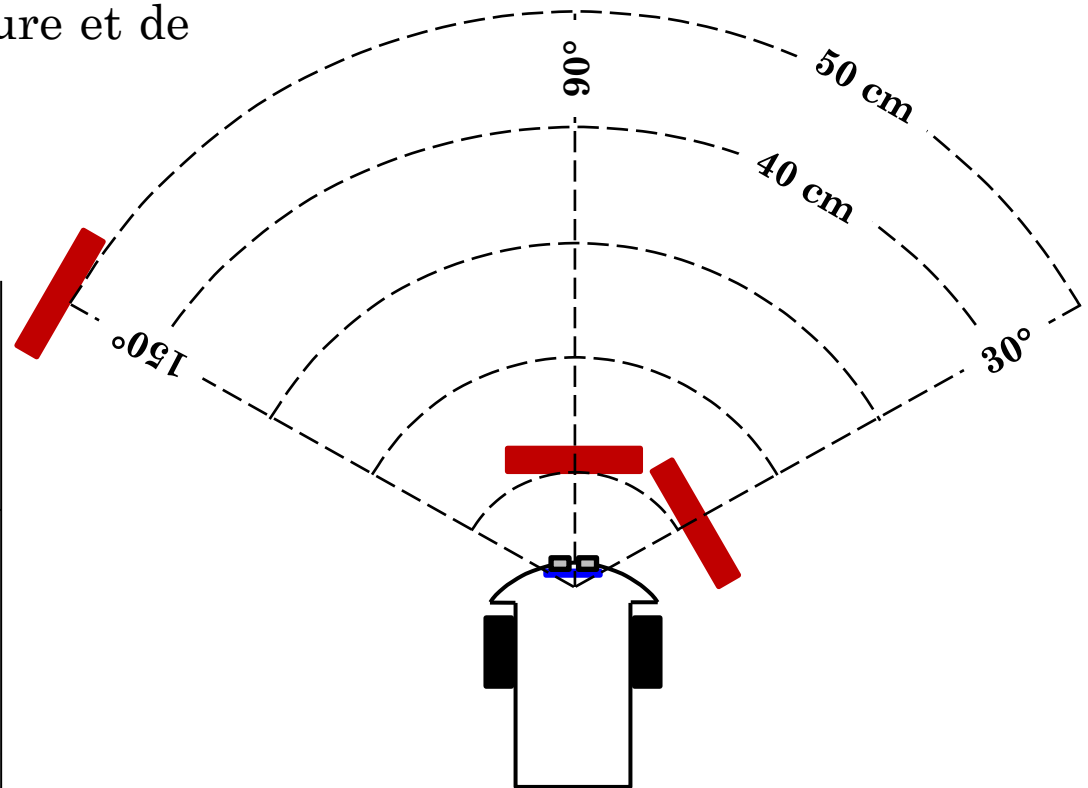


## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- Elle correspond au fonctionnement que l'on souhaite avoir pour la voiture en fonction de l'angle de mesure et de la distance des obstacles
- On définit plusieurs actions

150°	90°	30°	G	D
50	50	50	1	1
20	50	20	1	1
50	10	50	0	1
10	10	10	0	1
50	10	10		

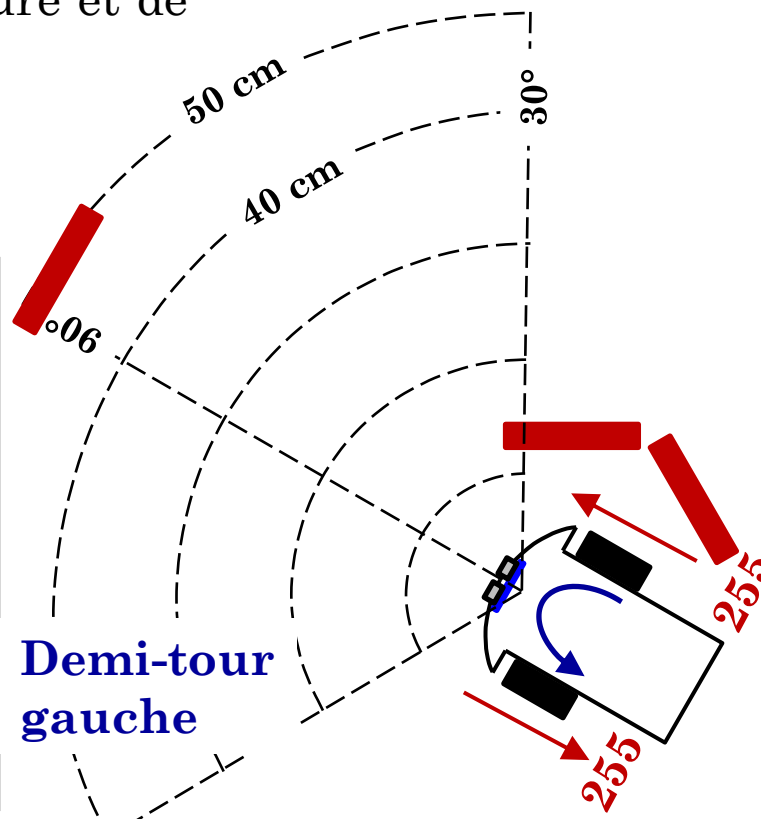


## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- Elle correspond au fonctionnement que l'on souhaite avoir pour la voiture en fonction de l'angle de mesure et de la distance des obstacles
- On définit plusieurs actions

150°	90°	30°	G	D
50	50	50	1	1
20	50	20	1	1
50	10	50	0	1
10	10	10	0	1
50	10	10	0	1

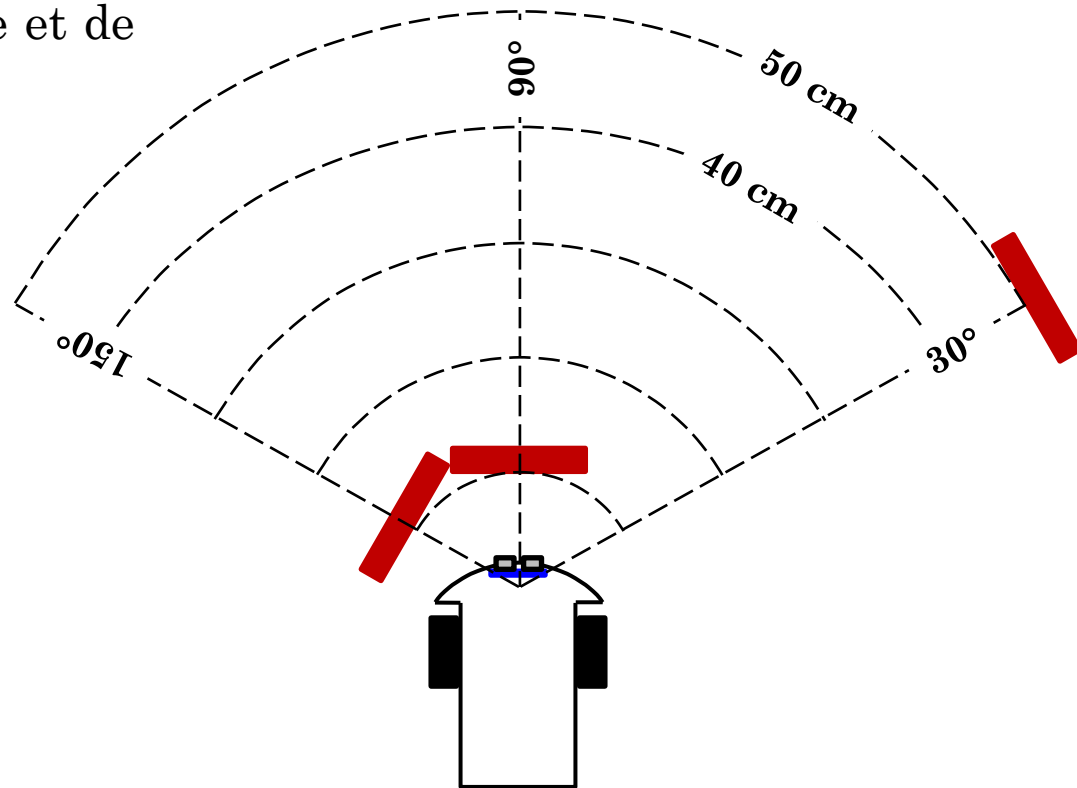


## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- Elle correspond au fonctionnement que l'on souhaite avoir pour la voiture en fonction de l'angle de mesure et de la distance des obstacles
- On définit plusieurs actions

150°	90°	30°	G	D
50	50	50	1	1
20	50	20	1	1
50	10	50	0	1
10	10	10	0	1
50	10	10	0	1
10	10	50		



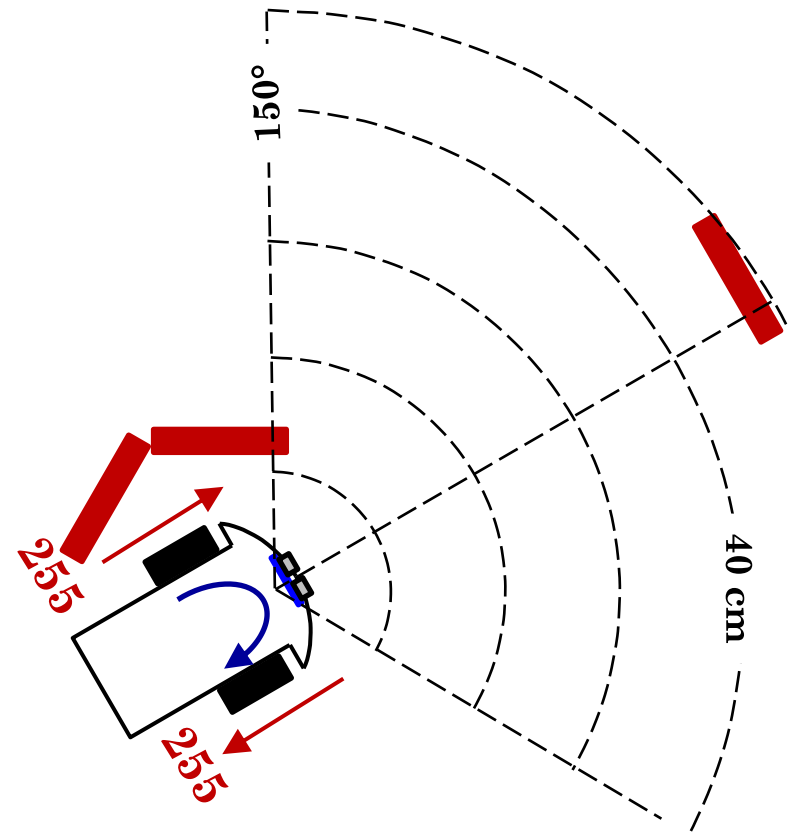
## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Base d'apprentissage

- Elle correspond au fonctionnement que l'on souhaite avoir pour la voiture en fonction de l'angle de mesure et de la distance des obstacles
- On définit plusieurs actions

150°	90°	30°	G	D
50	50	50	1	1
20	50	20	1	1
50	10	50	0	1
10	10	10	0	1
50	10	10	0	1
10	10	50	1	0

**Demi-tour  
droite**





## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Réseau 3-6-2

- On reste sur 4000 utilisations de la base pour calculer les poids et bias
- Une fois les calculs finis, on vérifie le résultat obtenu sur la base d'apprentissage (Gr et Dr)
- Les valeurs sont proches de 0 et 1 comme attendu.

150°	90°	30°	G	D	Gr	Dr
50	50	50	1	1	0.962	0.975
20	50	20	1	1	0.962	0.962
50	10	50	0	1	0.047	0.975
10	10	10	0	1	0.045	0.951
50	10	10	0	1	0.001	1
10	10	50	1	0	0.959	0.069

## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Réseau 3-12-2

- On reste sur 4000 utilisations de la base pour calculer les poids et bias
- L'ajout de 6 neurones (12 au total) dans la couche cachée n'améliore pas notablement le résultat

					3-6-2		3-12-2	
150°	90°	30°	G	D	Gr	Dr	Gr	Dr
50	50	50	1	1	0.962	0.975	0.966	0.974
20	50	20	1	1	0.962	0.962	0.962	0.968
50	10	50	0	1	0.047	0.975	0.042	0.971
10	10	10	0	1	0.045	0.951	0.042	0.961
50	10	10	0	1	0.001	1	0	1
10	10	50	1	0	0.959	0.069	0.967	0.063

## 4.3. Mis en œuvre de la bibliothèque NeuralNetwork

### □ Réseau 3-12-12-2

- On reste sur 4000 utilisations de la base pour calculer les poids et bias
- L'ajout de 6 neurones dans la couche cachée n'améliore pas notablement le résultat par contre l'ajout d'une couche cachée de 12 neurones donne de très mauvais résultats (même avec 10000 utilisations de la base d'apprentissage).

					3-6-2		3-12-2		3-12-12-2	
150°	90°	30°	G	D	Gr	Dr	Gr	Dr	Gr	Dr
50	50	50	1	1	0.962	0.975	0.966	0.974	0.579	0.821
20	50	20	1	1	0.962	0.962	0.962	0.968	0.591	0.825
50	10	50	0	1	0.047	0.975	0.042	0.971	0.549	0.82
10	10	10	0	1	0.045	0.951	0.042	0.961	0.208	0.844
50	10	10	0	1	0.001	1	0	1	0.505	0.827
10	10	50	1	0	0.959	0.069	0.967	0.063	0.563	0.82

### 4.4. Test et corrections (réseau 3-6-2)

#### ❑ Approche n°1

- Elle consiste à alterner entre mesure des 3 distances et modification de la vitesse/sens des moteurs
- La principale difficulté ici est que la voiture continue de se déplacer durant les mesures ce qui :
  - ✓ Cause une différence au niveau de l'environnement entre le moment des mesures et le moment de la prise de décision
  - ✓ Donne des déplacements incohérents de la voiture
- On peut cependant apporter 2 solutions :
  - ✓ Diminuer considérablement la vitesse max de la voiture (passer de 255 à 150 en PWM par exemple)
  - ✓ Ne pas utiliser un servomoteur mais 3 capteurs de distance laser afin de considérablement réduire le temps total de mesure

### 4.4. Test et corrections (réseau 3-6-2)

#### □ Approche n°2

- Alternner deux phases
  - ✓ Mesure des distances, la voiture étant à l'arrêt.
  - ✓ Mouvement de la voiture durant un certain temps, 500 ms par exemple.
- Pour cette approche, le déplacement de la voiture est saccadé.

## 5.1. Introduction

- La reconnaissance de lettre (ou de forme) est probablement l'application la plus marquante de l'IA.
- Ce qu'il est possible de faire avec la carte arduino ne peut en rien concurrencer les logiciels du commerce.
- L'application restera donc modeste pour 2 raisons :
  - ✓ La carte arduino n'est pas performante en termes de capacité mémoire et de vitesse de calcul comparé à un ordinateur.
  - ✓ La carte arduino n'est pas vraiment adaptée à l'utilisation de caméras.
- En conséquence, les images auront des dimensions réduites ainsi que le réseau de neurones.

## 5.2. Les contraintes

- La première des contraintes est la taille du réseau de neurones que peut accepter la carte arduino.
- En effet la génération de la matrice des poids et des bias peut rapidement prendre une place considérable en mémoire, de plus ce sont des nombres à virgule (codé sur 32 bits).
- En pratique, la déclaration d'une image de  $8 \times 8 = 64$  pixels (ou neurones d'entrée) avec une couche cachée de 32 neurones (nombre choisi de façon empirique) et un neurone de sortie fait planter la carte arduino MEGA.
- Par contre le programme se lance pour de  $8 \times 6 = 48$  pixels (ou neurones d'entrée) avec une couche cachée de 24 neurones et un neurone de sortie.

## 5.2. Les contraintes

- Vu la taille du réseau de neurone, il n'est pas envisageable d'utiliser une carte arduino UNO pour ce chapitre.
- Une dernière chose à prendre en compte est la base d'apprentissage. La librairie « exige » que les nombres soient des flottants. Là aussi, une image de « grande » taille nécessite l'utilisation de beaucoup d'espace mémoire.
- Pour résoudre ce problème, on peut envisager de modifier la librairie pour que la matrice d'entrée puisse être par exemple constituée de byte.
- Dans ce cas, et si on utilise uniquement du noir et blanc, une image de  $8 \times 4$  pixels peut être codée avec un nombre de type `uint32_t`. Le gain en espace mémoire est alors considérable.



## 5.3. Identification de la lettre « A »

### □ La base d'apprentissage

- On utilise la lettre A, en noir (1) et blanc (0) pour ce chapitre.
- Avec une image de 8x4 pixels, on peut choisir cette forme pour la lettre A.

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	1	0	0	1
8	0	0	0	0

image1

## 5.3. Identification de la lettre « A »

### □ La base d'apprentissage

- On utilise la lettre A, en noir (1) et blanc (0) pour ce chapitre.
- Avec une image de 8x4 pixels, on peut choisir cette forme pour la lettre A.
- En fonction de ce que la caméra va capter, la lettre peut se déplacer dans l'image.

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	1	0	0	1
8	0	0	0	0

image1

	1	2	3	4
1	1	1	1	1
2	1	0	0	1
3	1	0	0	1
4	1	1	1	1
5	1	0	0	1
6	1	0	0	1
7	0	0	0	0
8	0	0	0	0

image2

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	1	1	1	1
4	1	0	0	1
5	1	0	0	1
6	1	1	1	1
7	1	0	0	1
8	1	0	0	1

image3

## 5.3. Identification de la lettre « A »

### □ La base d'apprentissage

- On utilise la lettre A, en noir (1) et blanc (0) pour ce chapitre.
- Avec une image de 8x4 pixels, on peut choisir cette forme pour la lettre A.
- En fonction de ce que la caméra va capter, la lettre peut se déplacer dans l'image.
- La lettre A peut même avoir des formes différentes (sans parler de son orientation).
- Pour ces images, le neurone de sortie doit donner la valeur « 1 ».

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	1	0	0	1
8	0	0	0	0

image1

	1	2	3	4
1	1	1	1	1
2	1	0	0	1
3	1	0	0	1
4	1	1	1	1
5	1	0	0	1
6	1	0	0	1
7	0	0	0	0
8	0	0	0	0

image2

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	1	1	1	1
4	1	0	0	1
5	1	0	0	1
6	1	1	1	1
7	1	0	0	1
8	1	0	0	1

image3

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	1	1	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	0	0	0	0
8	0	0	0	0

image5

	1	2	3	4
1	0	0	0	0
2	0	1	1	1
3	0	1	0	1
4	0	1	0	1
5	0	1	1	1
6	0	1	0	1
7	0	1	0	1
8	0	0	0	0

image6

## 5.3. Identification de la lettre « A »

### □ La base d'apprentissage

- Voici des exemples d'images qui ne correspondent pas à la lettre A (absence de lettre, lettre très déformée, autre lettre ou signe).
- Dans ce cas, le neurone de sortie donnera la valeur 0.

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0

image7

	1	2	3	4
1	0	0	0	0
2	1	0	0	0
3	1	0	0	0
4	1	0	0	0
5	1	1	1	1
6	0	0	0	1
7	0	0	0	1
8	0	0	0	0

image8

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	0	0	0	1
5	0	1	1	1
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0

image13

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	1	1	1	1
8	0	0	0	0

image16

## 5.3. Identification de la lettre « A »

### ❑ Réseau 32-16-1: calcul des poids et bias

- On choisi la base suivante pour l'apprentissage :

```
const float entrees [8][32] = {  
  {0,0,0,0,1,1,1,1,1,0,0,1,1,0,0,1,1,1,1,1,0,0,1,1,0,0,1,0,0,0,0}, //image1  
  {1,1,1,1,1,0,0,1,1,0,0,1,1,1,1,1,0,0,1,1,0,0,1,0,0,0,0,0,0,0}, //image2  
  {0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,1,1,0,0,1,1,1,1,1,1,0,0,1,1,0,0,1}, //image3  
  {0,0,0,0,0,1,1,1,0,1,0,1,0,1,0,1,0,1,1,1,0,1,0,1,0,1,0,1,0,0,0,0}, //image6  
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}, //image7  
  {0,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,1,1,1,0,0,0,1,0,0,0,1,0,0,0,0}, //image8  
  {0,0,0,0,1,1,1,1,1,0,0,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0}, //image13  
  {0,0,0,0,1,1,1,1,1,0,0,1,1,0,0,1,1,1,1,1,1,0,0,1,1,1,1,1,0,0,0,0}, //image16  
};
```

```
const float sortiesAttendues [8] [1] = {  
  {1}, {1}, {1}, {1}, {0}, {0}, {0}, {0},  
};
```

## 5.3. Identification de la lettre « A »

### □ Réseau 32-16-1: calcul des poids et bias

- Le calcul des  $32 \times 16 + 16 \times 1 = 528$  poids et des 2 bias a pris 19 mn et 40 s pour 4000 utilisations de la base d'apprentissage.
- Les valeurs, de la sortie, calculées ( $S_r$ ) pour les 8 lignes (images) de la base d'apprentissage sont très proches des valeurs désirées ( $S$ ).

S	$S_r$
1	0.989
1	0.996
1	0.998
1	0.995
0	0.007
0	0.004
0	0.006
0	0.008

## 5.3. Identification de la lettre « A »

### □ Réseau 32-16-1: identification

- Pour vérifier l'apprentissage, on utilise des images supplémentaires et on calcule la valeur de la sortie du réseau de neurones.

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	1	1	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	0	0	0	0
8	0	0	0	0

image5

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	0	0	0	1
5	1	1	1	1
6	1	0	0	1
7	1	0	0	0
8	0	0	0	0

image10

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	0	0	1
6	1	0	0	1
7	1	1	1	1
8	0	0	0	0

image12

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0

image14

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	1	1	1	1
8	0	0	0	0

image16

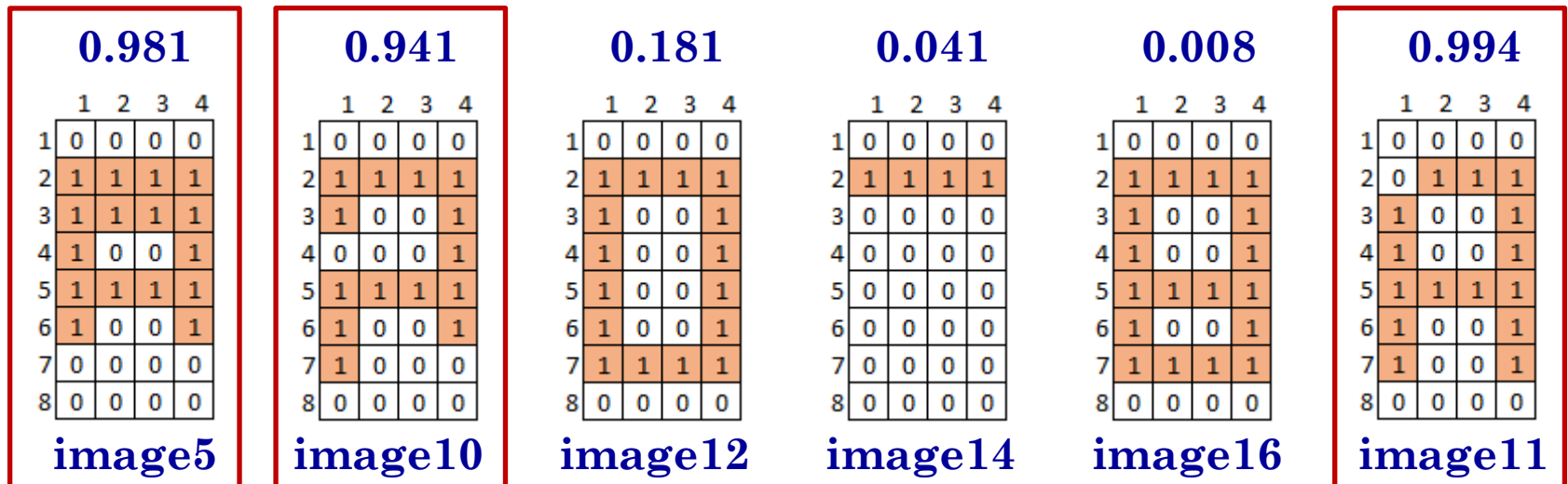
	1	2	3	4
1	0	0	0	0
2	0	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	1	0	0	1
8	0	0	0	0

image11

## 5.3. Identification de la lettre « A »

### □ Réseau 32-16-1: identification

- Pour vérifier l'apprentissage, on utilise des images supplémentaires et on calcule la valeur de la sortie du réseau de neurones.
- Lorsque la sortie est supérieure à 0.5, le réseau de neurone a reconnu la lettre A, et vous ?





## 5.3. Identification de la lettre « A »

### □ Réseau 32-16-1: identification

- Regardons, à présent, la valeur de la sortie quand on enlève des pixels à la lettre A.

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	0	0	0	1
8	0	0	0	0

image21

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	0
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	0	0	0	1
8	0	0	0	0

image22

	1	2	3	4
1	0	0	0	0
2	1	0	1	1
3	1	0	0	0
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	0	0	0	1
8	0	0	0	0

image23

	1	2	3	4
1	0	0	0	0
2	1	0	1	1
3	0	0	0	0
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	0	0	0	1
8	0	0	0	0

image24

	1	2	3	4
1	0	0	0	0
2	1	0	1	1
3	0	0	0	0
4	1	0	0	1
5	1	1	1	1
6	1	0	0	0
7	0	0	0	1
8	0	0	0	0

image25

	1	2	3	4
1	0	0	0	0
2	1	0	1	1
3	0	0	0	0
4	1	0	0	1
5	0	1	1	1
6	1	0	0	0
7	0	0	0	1
8	0	0	0	0

image26

## 5.3. Identification de la lettre « A »

### ❑ Réseau 32-16-1: identification

- Regardons, à présent, la valeur de la sortie quand on enlève des pixels à la lettre A.
- Le réseau arrive à identifier la lettre A dans les images 21, 22 et 24. Il n'en trouve pas dans les images 25 et 26.
- Le résultat pour les images 23 et 24 n'est pas compréhensible à priori car il y a un pixel de moins dans l'image 24.

**0.938**

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	0	0	0	1
8	0	0	0	0

**image21**

**0.506**

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	0
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	0	0	0	1
8	0	0	0	0

**image22**

**0.291**

	1	2	3	4
1	0	0	0	0
2	1	0	1	1
3	1	0	0	0
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	0	0	0	1
8	0	0	0	0

**image23**

**0.553**

	1	2	3	4
1	0	0	0	0
2	1	0	1	1
3	0	0	0	0
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	0	0	0	1
8	0	0	0	0

**image24**

**0.075**

	1	2	3	4
1	0	0	0	0
2	1	0	1	1
3	0	0	0	0
4	1	0	0	1
5	1	1	1	1
6	1	0	0	0
7	0	0	0	1
8	0	0	0	0

**image25**

**0.029**

	1	2	3	4
1	0	0	0	0
2	1	0	1	1
3	0	0	0	0
4	1	0	0	1
5	0	1	1	1
6	1	0	0	0
7	0	0	0	1
8	0	0	0	0

**image26**

## 5.3. Identification de la lettre « A »

### □ Réseau 32-16-1: identification

- Pour finir, nous pouvons regarder, l'impact de la taille et de l'orientation de la lettre sur son identification.

0.989				
	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	1	0	0	1
8	0	0	0	0
image1				

	1	2	3	4
1	0	0	0	0
2	1	1	1	0
3	1	0	1	0
4	1	1	1	0
5	1	0	1	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
image27				

	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	1	1	1
7	1	0	0	1
8	1	0	0	1
image28				

	1	2	3	4
1	0	0	0	0
2	1	0	0	1
3	1	0	0	1
4	1	1	1	1
5	1	0	0	1
6	1	0	0	1
7	1	1	1	1
8	0	0	0	0
image29				

## 5.3. Identification de la lettre « A »

### □ Réseau 32-16-1: identification

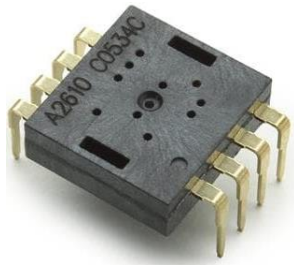
- Pour finir, nous pouvons regarder, l'impact de la taille et de l'orientation de la lettre sur son identification.
- Le réseau arrive à reconnaître la lettre A si elle est plus grande ou plus petite. Par contre il n'arrive pas à lire à l'envers car on ne lui a pas appris à le faire.

	0.989				0.659				0.999				0.395			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
2	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	1
3	1	0	0	1	1	0	1	0	1	0	0	1	1	0	0	1
4	1	0	0	1	1	1	1	0	1	0	0	1	1	1	1	1
5	1	1	1	1	1	0	1	0	1	1	1	1	1	0	0	1
6	1	0	0	1	0	0	0	0	1	1	1	1	1	0	0	1
7	1	0	0	1	0	0	0	0	1	0	0	1	1	1	1	1
8	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
	image1				image27				image28				image29			

## 5.4. Capture d'une image et identification

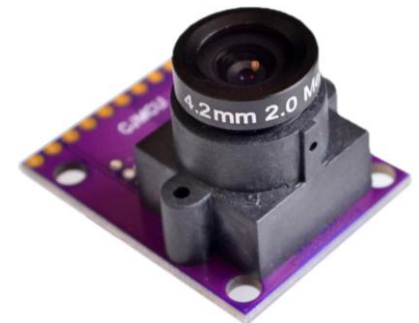
### □ Présentation du capteur

- Une possibilité pour obtenir une image avec peu de pixels, et d'utiliser une caméra de souris optique qui peut être pilotée avec une carte arduino.



- l'ADNS-2610 est une caméra 18×18 pixels qui n'est malheureusement pas montée avec une optique, à part celle de la souris qui oblige le capteur à être quasi collé « au bureau ».

- l'ADNS-3080 est une caméra 30×30 pixels que l'on peut trouver montée avec une optique ce qui permet de s'éloigner de l'image à capturer.

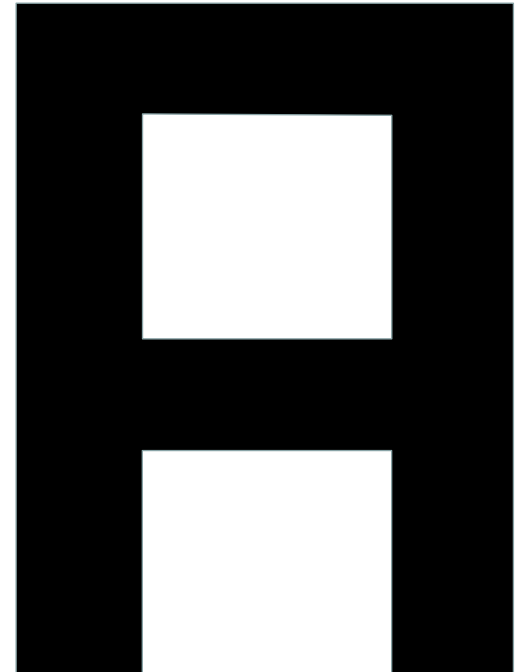


- Pour plus d'information sur ces capteurs et la prise d'images en général, vous pouvez regarder le cours « [Eléments de robotique avec arduino : Caméras et images](#) »

## 5.4. Capture d'une image et identification

### □ Capture d'une image

- La lettre A est imprimée en respectant les dimensions de l'image n°1 de la base d'apprentissage.



## 5.4. Capture d'une image et identification

### □ Capture d'une image

- La lettre A est imprimée en respectant les dimensions de l'image n°1 de la base d'apprentissage.
- La distance et la focale ont été réglées pour que les barres montantes de la lettre aient une largeur de 4 carreaux.
- Certaines aberrations apparaissent sur l'image.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
12	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
13	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
14	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
15	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
19	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
20	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
21	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
22	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
23	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
24	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
25	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
26	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## 5.4. Capture d'une image et identification

### □ Capture d'une image

- La lettre A est imprimée en respectant les dimensions de l'image n°1 de la base d'apprentissage.
- La distance et la focale ont été réglées pour que les barres montantes de la lettre aient une largeur de 4 carreaux.
- Certaines aberrations apparaissent sur l'image.
- Il faut ramener la lettre à une image de 8x4 pixels.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
19	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
20	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
21	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
22	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



## 5.4. Capture d'une image et identification

### □ Extraction de la lettre – la méthode

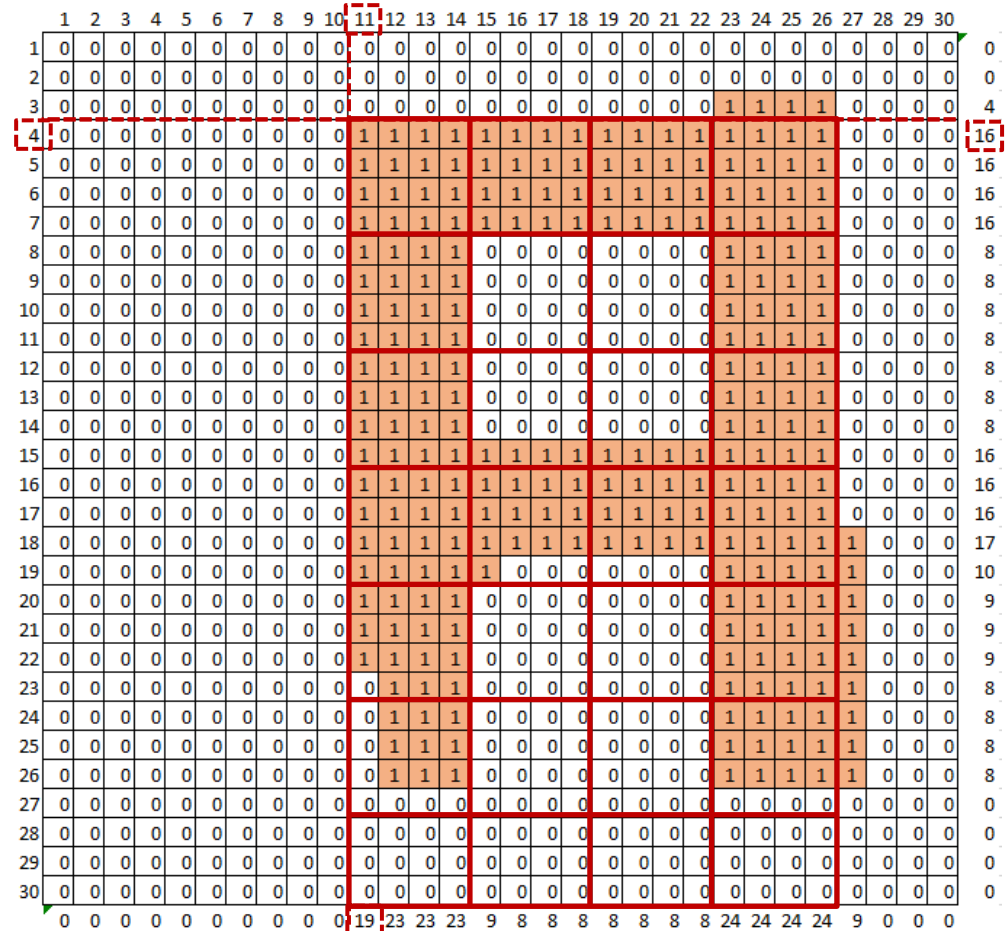
- On commence par faire un histogramme du chiffre 1 pour les lignes et les colonnes.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30		
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	4	
4	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	16	
5	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	16	
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	16	
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	16	
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	8	
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	8	
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	8	
11	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	8	
12	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	8	
13	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	8	
14	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	8	
15	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	16	
16	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	16	
17	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	16	
18	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	17
19	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	10
20	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	9
21	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	9
22	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	9
23	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	8
24	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	8
25	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	8
26	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	8
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	19	23	23	23	9	8	8	8	8	8	8	8	8	24	24	24	24	9	0	0	0	0

## 5.4. Capture d'une image et identification

### □ Extraction de la lettre – la méthode

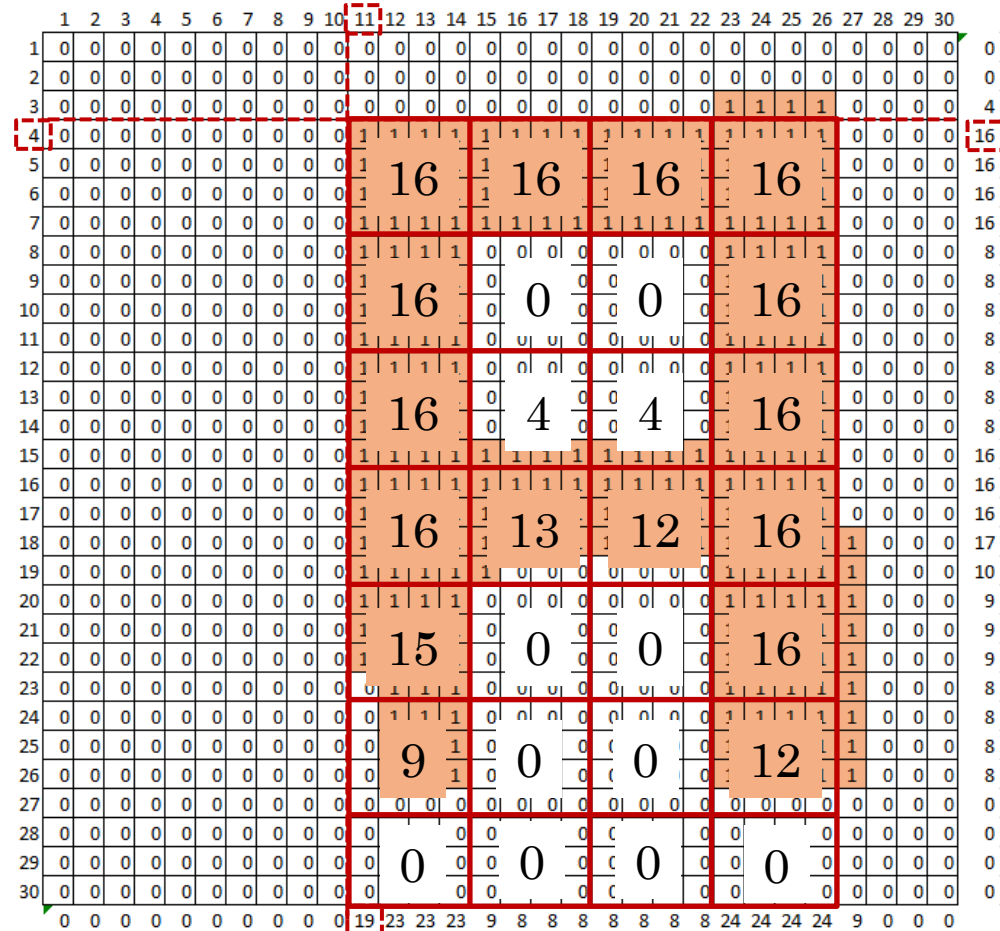
- On commence par faire un histogramme du chiffre 1 pour les lignes et les colonnes.
- On repère le démarrage de la lettre en se fixant un nombre de 1 sur une même ligne ou colonne par exemple supérieur à 6.



## 5.4. Capture d'une image et identification

### □ Extraction de la lettre – la méthode

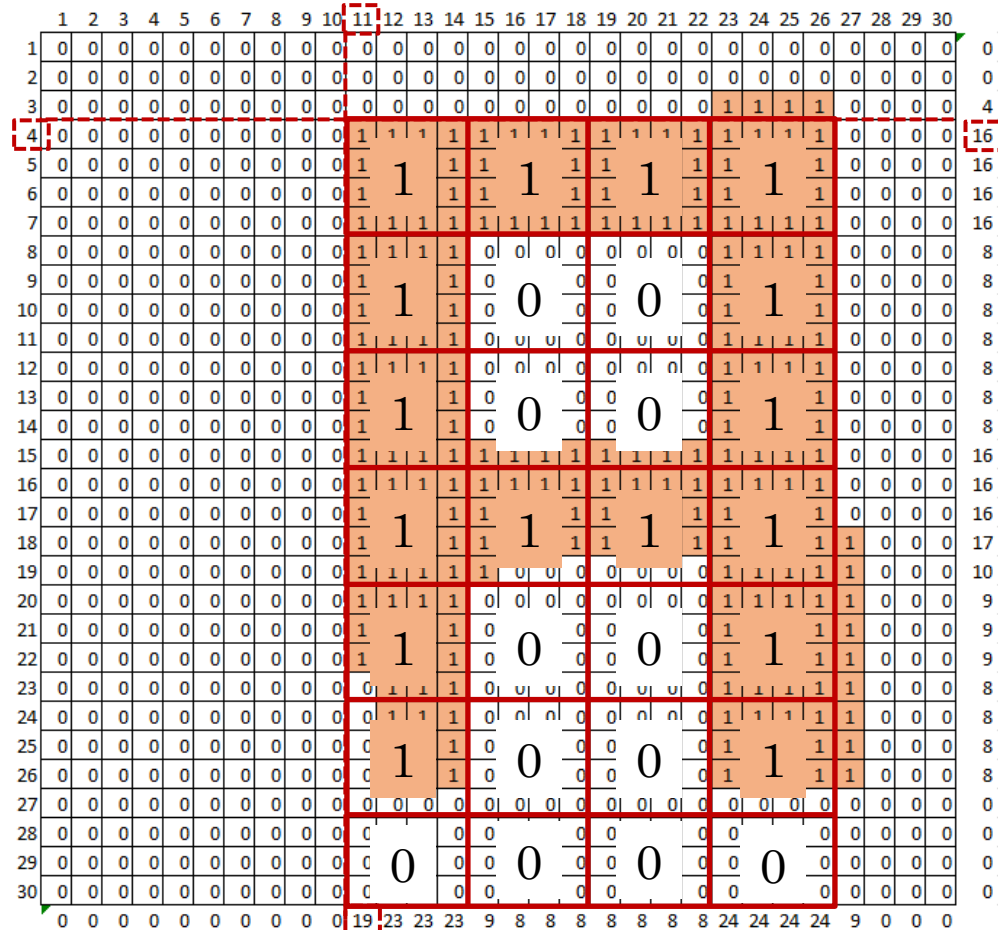
- On commence par faire un histogramme du chiffre 1 pour les lignes et les colonnes.
- On repère le démarrage de la lettre en se fixant un nombre de 1 sur une même ligne ou colonne par exemple supérieur à 6.
- On additionne les « 1 » dans chaque sous image de 4×4 pixels



## 5.4. Capture d'une image et identification

### ❑ Extraction de la lettre – la méthode

- On commence par faire un histogramme du chiffre 1 pour les lignes et les colonnes.
- On repère le démarrage de la lettre en se fixant un nombre de 1 sur une même ligne ou colonne par exemple supérieur à 6.
- On additionne les « 1 » dans chaque sous image de 4×4 pixels
- Pour la nouvelle image 8×4 pixels, on considère qu'on a un « 1 » si la somme est supérieure ou égale à 8 sinon la valeur est « 0 »



## 5.4. Capture d'une image et identification

### □ Extraction de la lettre – la méthode

- On obtient finalement l'image  
8×4 pixels que le réseau de  
neurones va pouvoir analyser.

1	1	1	1
1	0	0	1
1	0	0	1
1	1	1	1
1	0	0	1
1	0	0	1
0	0	0	0

## 5.4. Capture d'une image et identification

### □ Extraction de la lettre – la méthode

- On obtient finalement l'image 8×4 pixels que le réseau de neurones va pouvoir analyser.
- A noter que la 8<sup>ème</sup> ligne de la nouvelle image n'a pas pu être traitée donc par défaut chaque pixel de cette ligne est à « 0 ».
- On retrouve ici l'image n°2 de la base d'apprentissage.

1	1	1	1
1	0	0	1
1	0	0	1
1	1	1	1
1	0	0	1
1	0	0	1
0	0	0	0
0	0	0	0

## 5.4. Capture d'une image et identification

### □ Extraction de la lettre – le programme

```
for (i=0; i<8; i++) {  
    for (j=0; j<4; j++) {  
        image [i] [j]=0; } }
```

} Initialisation des pixels de l'image 8×4

```
pixel=0;  
demar_lignes = 0;  
demar_colonnes = 0;  
num_lignes = 0;  
num_colonnes = 0;
```

} Initialisation des variables

```
for (i=0; i<30; i++) {  
    for (j=0; j<30; j++) {  
        num_lignes = num_lignes + imageADNS [i] [j];  
        num_colonnes = num_colonnes + imageADNS [j] [i];  
        lignes [i] = num_lignes;  
        colonnes [i] = num_colonnes;}
```

} Histogramme sur les  
lignes et colonnes  
de l'image donnée  
par le capteur  
ADNS3080

## 5.4. Capture d'une image et identification

### ❑ Extraction de la lettre – le programme

```
i=0;
compteur = 0;
while ((compteur==0)*(i<30)) {
    if (lignes [i]>6) {
        demar_lignes = i;
        compteur = 1;}
    i++;
}
```

**Recherche du numéro de la ligne qui correspond au début de la lettre**

```
i=0;
compteur = 0;
while ((compteur==0)*(i<30)) {
    if (colonnes [i]>6) {
        demar_colonnes = i;
        compteur = 1;}
    i++;
}
```

**Recherche du numéro de la colonne qui correspond au début de la lettre**



## 5.4. Capture d'une image et identification

### ❑ Extraction de la lettre – le programme

- Il reste alors à déterminer si la valeur de chaque pixel est 1 ou 0.

```
for (i=0; i<6; i++) {  
  for (j=0; j<4; j++) {  
    pixel = 0;  
    for (ik=0; ik<4; ik++) {  
      for (jk=0; jk<4; jk++) {  
        pixel=pixel+imageADNS[i*4+demar_lignes+ik] [j*4+demar_colonnes+jk];  
      }  
    }  
    if (pixel > 7) {image [i] [j] = 1;}  
  }  
}
```

## 5.4. Capture d'une image et identification

### □ Identification en pratique

- Le système fonctionne comme prévu, cependant certaines images sont reconnues comme une lettre A alors que cela ne devrait pas être le cas.

- C'est notamment le cas de cette image :

	1	2	3	4
1	1	0	0	0
2	1	0	1	0
3	1	0	0	1
4	1	0	0	1
5	0	0	0	0
6	1	1	1	1
7	0	0	0	0
8	0	0	0	0

**image31**

- Dans ce cas, il faut intégrer cette image à la base d'apprentissage pour réajuster les poids.

## 5.4. Capture d'une image et identification

### Identification en pratique

- La base passe de 8 à 14 images avec un temps de calcul des poids de 35mn.

**A**

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	1	0	0	1
8	0	0	0	0

image1

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	1	1	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	0	0	0	0
8	0	0	0	0

image5

	1	2	3	4
1	1	1	1	1
2	1	0	0	1
3	1	0	0	1
4	1	1	1	1
5	1	0	0	1
6	1	0	0	1
7	0	0	0	0
8	0	0	0	0

image2

	1	2	3	4
1	0	0	0	0
2	0	1	1	1
3	0	1	0	1
4	0	1	0	1
5	0	1	1	1
6	0	1	0	1
7	0	1	0	1
8	0	0	0	0

image6

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	1	1	1	1
4	1	0	0	1
5	1	0	0	1
6	1	1	1	1
7	1	0	0	1
8	1	0	0	1

image3

	1	2	3	4
1	0	0	0	0
2	1	1	1	0
3	1	0	1	0
4	1	1	1	0
5	1	0	1	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0

image27

**Pas A**

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0

image7

	1	2	3	4
1	0	0	0	0
2	0	0	1	1
3	0	1	0	1
4	1	0	0	1
5	0	0	0	1
6	0	0	0	1
7	0	0	0	1
8	0	0	0	0

image20

	1	2	3	4
1	0	0	0	0
2	1	0	0	0
3	1	0	0	0
4	1	0	0	0
5	1	1	1	1
6	0	0	0	1
7	0	0	0	1
8	0	0	0	0

image8

	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	1	1	1
6	1	1	1	1
7	1	1	1	1
8	1	1	1	1

image4

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	0	0	0	1
5	0	1	1	1
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0

image13

	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	1	1	1
6	1	1	1	1
7	0	0	0	0
8	0	0	0	0

image30

	1	2	3	4
1	0	0	0	0
2	1	1	1	1
3	1	0	0	1
4	1	0	0	1
5	1	1	1	1
6	1	0	0	1
7	1	1	1	1
8	0	0	0	0

image16

	1	2	3	4
1	1	0	0	0
2	1	0	1	0
3	1	0	0	1
4	1	0	0	1
5	0	0	0	0
6	1	1	1	1
7	0	0	0	0
8	0	0	0	0

image31

## 5.4. Capture d'une image et identification

### □ Identification en pratique

- Avec cette base, le réseau de neurone arrive aussi à reconnaître l'image 29 comme étant la lettre A .
- Le changement de valeur des poids et bias a aussi un impact sur l'identification de la série suivante.
- A présent, l'image 24 n'est plus reconnue.

	1	2	3	4
1	0	0	0	0
2	1	0	0	1
3	1	0	0	1
4	1	1	1	1
5	1	0	0	1
6	1	0	0	1
7	1	1	1	1
8	0	0	0	0

image29

0.943	0.606	0.080	0.143	0.109	0.000
1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4
1 0 0 0 0	1 0 0 0 0	1 0 0 0 0	1 0 0 0 0	1 0 0 0 0	1 0 0 0 0
2 1 1 1 1	2 1 1 1 1	2 1 0 1 1	2 1 0 1 1	2 1 0 1 1	2 1 0 1 1
3 1 0 0 1	3 1 0 0 0	3 1 0 0 0	3 0 0 0 0	3 0 0 0 0	3 0 0 0 0
4 1 0 0 1	4 1 0 0 1	4 1 0 0 1	4 1 0 0 1	4 1 0 0 1	4 1 0 0 1
5 1 1 1 1	5 1 1 1 1	5 1 1 1 1	5 1 1 1 1	5 1 1 1 1	5 0 1 1 1
6 1 0 0 1	6 1 0 0 1	6 1 0 0 1	6 1 0 0 1	6 1 0 0 0	6 1 0 0 0
7 0 0 0 1	7 0 0 0 1	7 0 0 0 1	7 0 0 0 1	7 0 0 0 1	7 0 0 0 1
8 0 0 0 0	8 0 0 0 0	8 0 0 0 0	8 0 0 0 0	8 0 0 0 0	8 0 0 0 0
image21	image22	image23	image24	image25	image26