



# Sciences et technologies de l'Industrie et du développement durable

## ET 24 : Modèle de comportement d'un système

### Labview et Lego MINDSTORM.

Sciences et Technologies de l'Industrie et du Développement Durable <i>Formation des enseignants</i>	
parcours : ET24	<b>Modèle de comportement d'un système</b>
<b>Durée</b> : 3 h.	
<b>Objectif</b> : Etre capable de réaliser le pilotage des E/S de la brique Lego NXT à l'aide de Labview.	
<b>Pré-requis</b> : Les bases de Labview	
<b>Bases théoriques</b> : Aucune.	
<b>Outil</b> : Labview ; kit lego Mindstorm.	
<b>Support</b> :	
<b>Modalités</b> : Activité sous forme de TD	
<b>Synthèse et validation</b> : Être capable de recréer en autonomie les modèles proposés.	



# Sciences et technologies

## de l'Industrie et du développement durable

### Sommaire

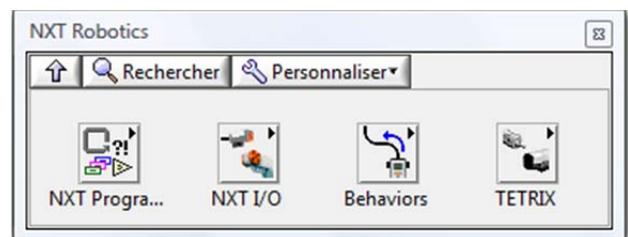
<b>1</b>	<b>Objectif.....</b>	<b>3</b>
<b>2</b>	<b>Présentation de la carte et de l'interface Labview.....</b>	<b>3</b>
2.1	Carte Arduino.....	3
2.2	Interface Labview.....	3
<b>3</b>	<b>Procédures élémentaires.....</b>	<b>4</b>
3.1	Sortie digitale : allumage de la led L.....	<i>Erreur ! Signet non défini.</i>
3.2	Entrée Analogique.....	<i>Erreur ! Signet non défini.</i>
3.3	Sortie analogique.....	<i>Erreur ! Signet non défini.</i>
<b>4</b>	<b>Pilotage des E/S d'une carte Arduino au moyen d'un Statechart.....</b>	<i>Erreur ! Signet non défini.</i>
<b>5</b>	<b>Exploitation des exemples.....</b>	<i>Erreur ! Signet non défini.</i>
<b>6</b>	<b>Pilotage d'un servomoteur.....</b>	<i>Erreur ! Signet non défini.</i>
<b>7</b>	<b>Procédure d'installation.....</b>	<b>15</b>
<b>8</b>	<b>Pour aller plus loin.....</b>	<i>Erreur ! Signet non défini.</i>



# Sciences et technologies de l'Industrie et du développement durable

## 1 Objectif.

A ce stade de la formation Labview, vous connaissez déjà de nombreuses procédures de programmation avec ce logiciel.



L'objectif de ce document est de vous présenter des techniques de programmation de la brique Lego NXT avec Labview.

## 2 Présentation de la brique Lego et de l'interface Labview.

### 2.1 Brique Lego.



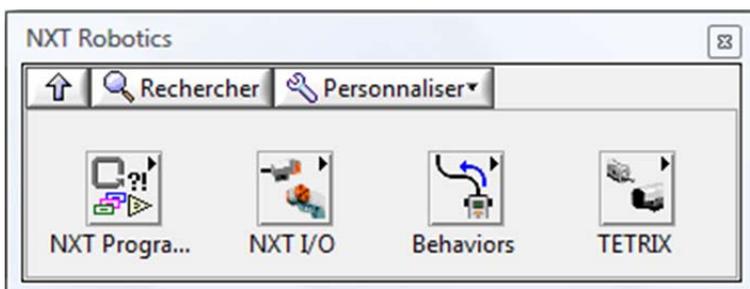
La brique Lego est composée :

- un microprocesseur 32 bits ARM7.
- 1 port USB 2.0.
- 4 ports d'entrée (1, 2, 3, 4)
- 3 ports de sortie (A, B, C)

Elle supporte la prise en charge de communications sans fil Bluetooth.

Elle s'alimente au moyen de 6 piles AA de 1,5 V, ou d'une batterie rechargeable.

### 2.2 Interface Labview.

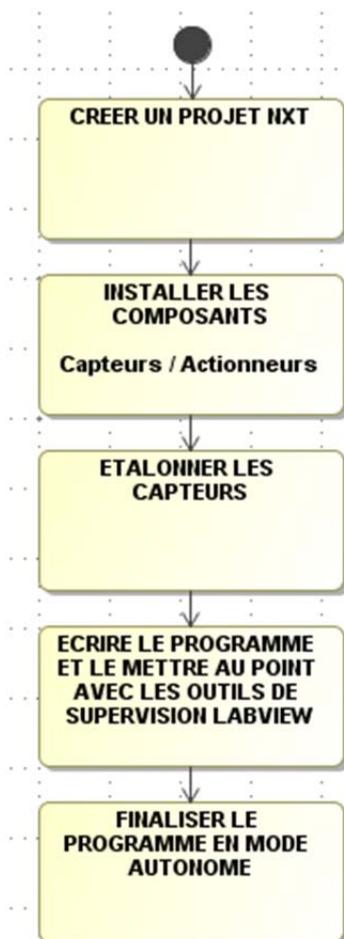


Une interface Labview est disponible. Elle s'insère comme une palette dans Labview, qui comporte uniquement les fonctions que l'on peut utiliser avec la brique NXT.



# Sciences et technologies de l'Industrie et du développement durable

## 3 Procédure de programmation.



Dans ce document, il est supposé que :

- l'interface Labview est déjà installée ;
- le comportement attendu du robot est déjà spécifié. Naturellement, au niveau STI2D, la spécification du comportement est écrite au moyen des diagrammes SysML !

La procédure de programmation est alors la suivante :

- 1) création d'un projet NXT dans Labview ;
- 2) installation des composants physique et test de ces composants ;
- 3) étalonnage des capteurs ;
- 4) écriture du programme, et test de ce programme en mode « connecté avec le PC », qui permet une supervision et un débogage efficace ;
- 5) finalisation du programme en mode autonome.

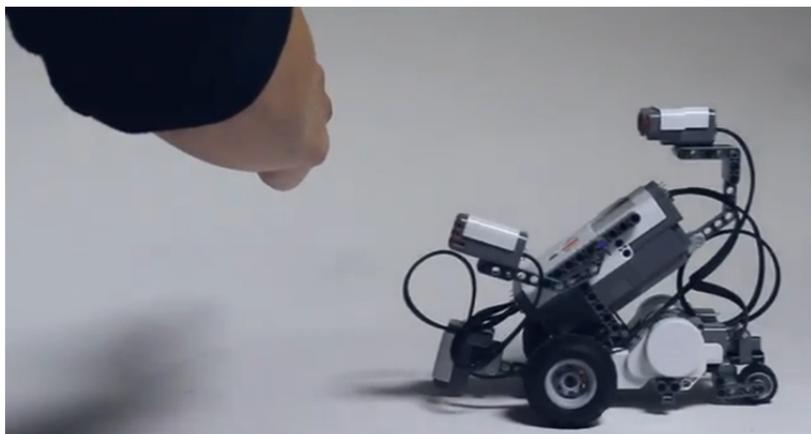
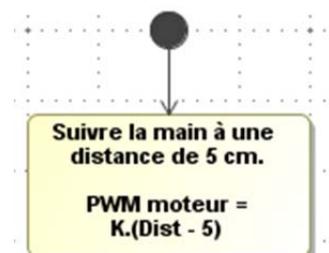
Cette procédure est illustrée par la programmation d'un comportement simple, le *Robot NXT suiveur* :

Dans cet exemple, nous allons programmer le robot pour qu'il suive la main à une distance de 5 cm.

Les moteurs du robot sont commandés en PWM.

Une commande de 100 correspond à un rapport cyclique de 1, et donc à une puissance maximale.

La commande de « puissance » envoyée au moteur sera donc égale à  $K \cdot (\text{Dist} - 5)$ , avec  $K$  une constante que vous déterminerez lors des test pour obtenir un fonctionnement acceptable.



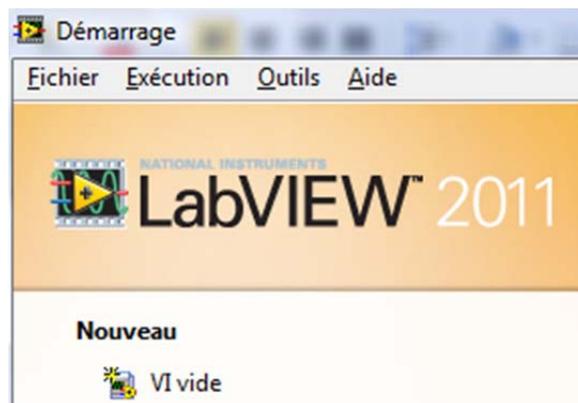


# Sciences et technologies de l'Industrie et du développement durable

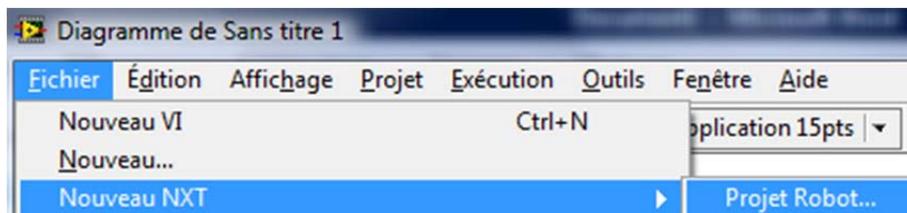
## 3.1 Créer un projet NXT.

Pour créer un nouveau projet NXT :

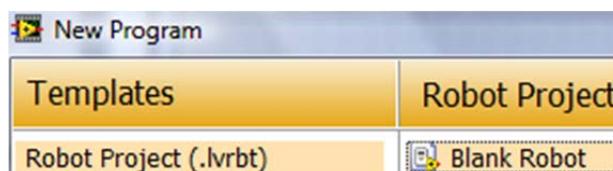
- Créer un nouveau VI vide :



- Créer un *Nouveau NXT / Projet Robot*. Cette commande lance l'interface de programmation NXT de Labview.



- Une fois cette interface lancée, créer un nouveau *Robot Project*, de type *Blank Robot*. (Blank = vierge)



- Enregistrer le projet là où vous le souhaitez.





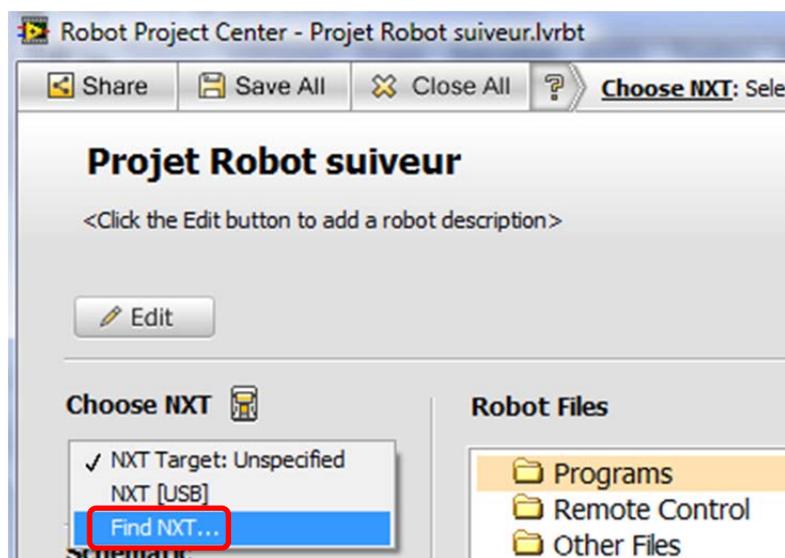
# Sciences et technologies de l'Industrie et du développement durable

## 3.2 Installer et tester les composants sur la brique Lego.

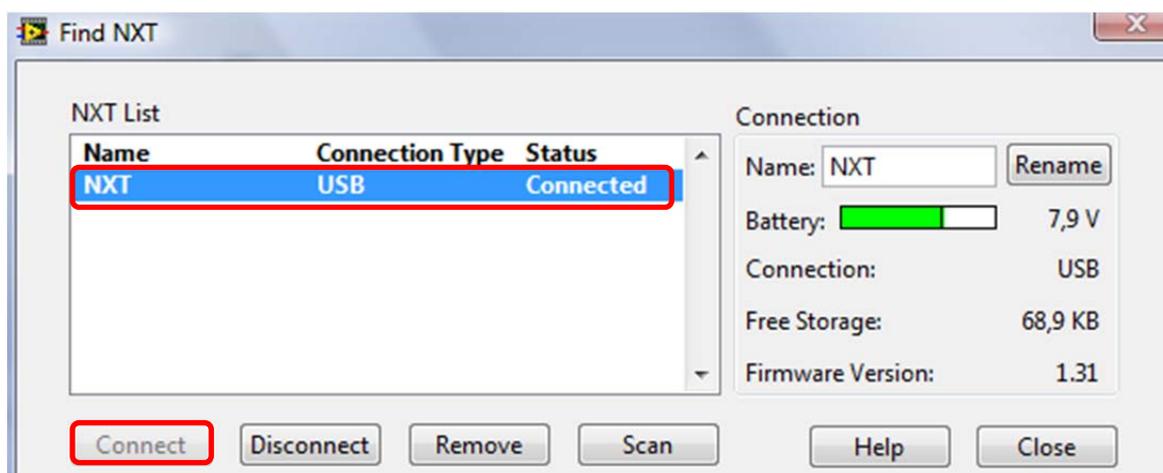
Labview peut vous aider à vérifier si les composants sont correctement installés sur la brique Lego.

Pour cela :

- branchez votre brique sur votre PC au moyen du port USB ;
- dans le Robot Project Center, lancez une recherche de votre brique NXT :



- si votre brique est correctement trouvée, Labview vous donne des informations sur ce composant :

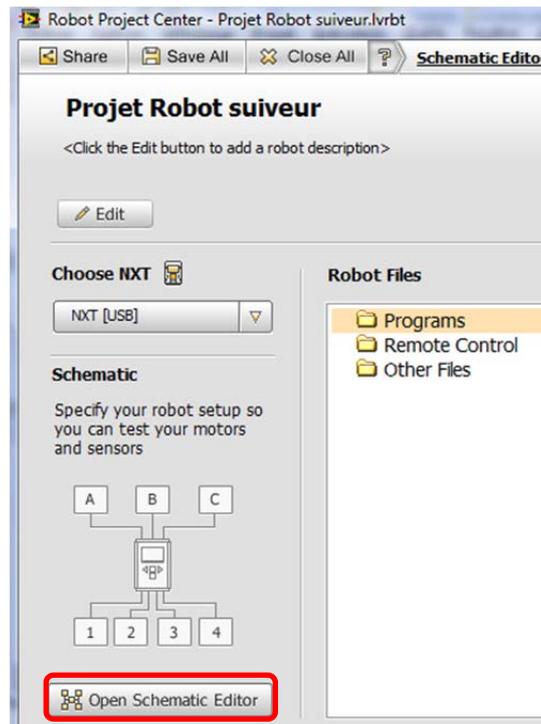


Si besoin, connectez-vous à cette brique au moyen du bouton *Connect*.

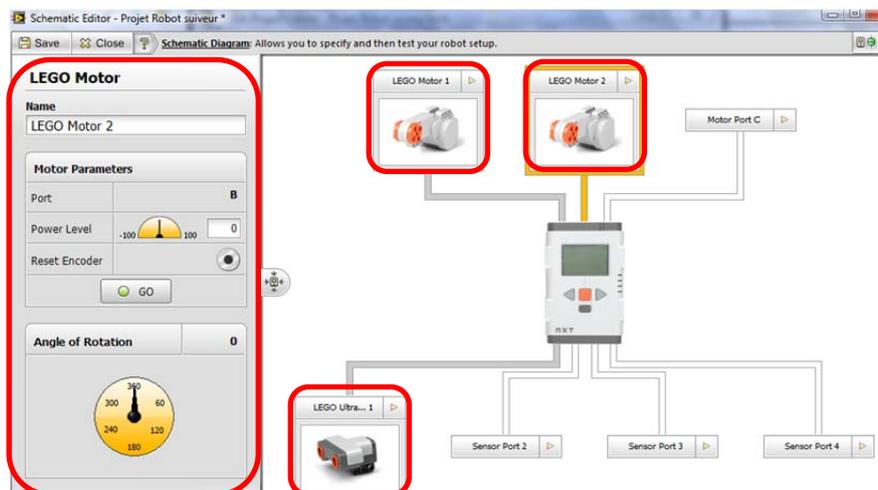


# Sciences et technologies de l'Industrie et du développement durable

- ensuite, ouvrez l'éditeur de schémas :



- puis déclarez les composants que vous branchez sur votre brique :



Dans notre cas :

- deux moteurs sur les ports A et B
- un capteur à ultra-son sur le port 1.

Utilisez l'interface intuitive sur la partie gauche de l'écran pour vérifier si vos composants fonctionnent.

Remarque : cette interface est uniquement une interface de test. Elle n'influence pas les programmes LV.



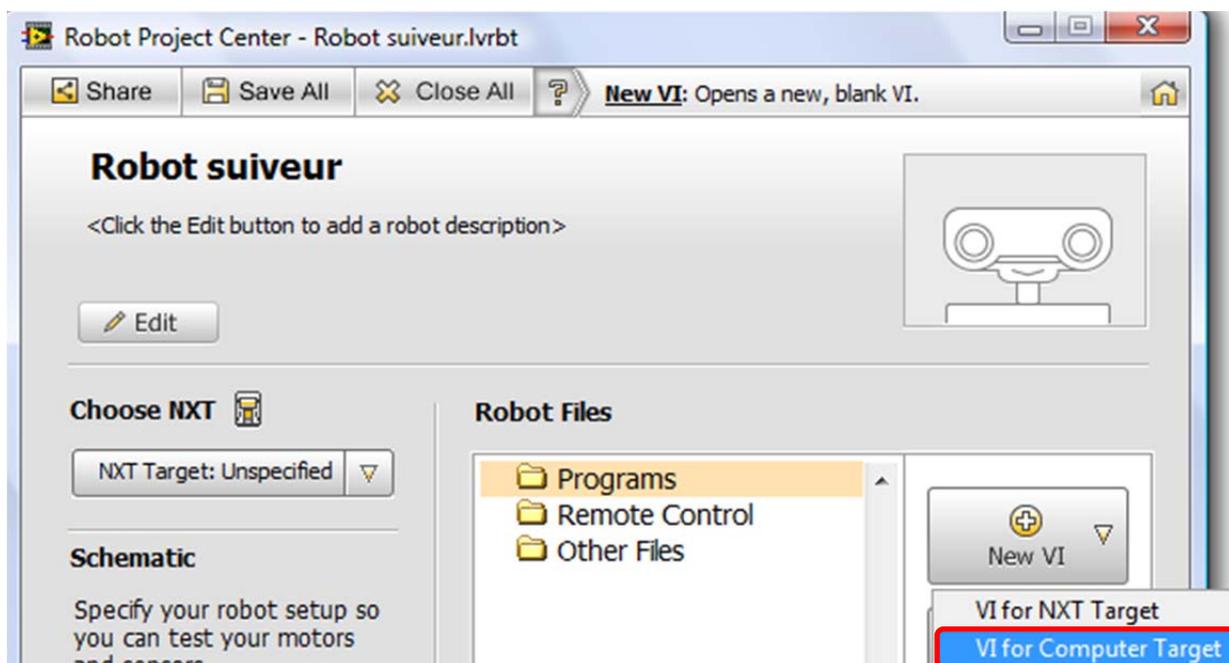
# Sciences et technologies de l'Industrie et du développement durable

## 3.3 Etalonnage des capteurs.

L'objectif est que le robot suive votre main à une distance de 5cm. Il faut donc connaître l'information renvoyée par le capteur à ultra-son dans cette configuration.

Il faut donc étudier le comportement du capteur à ultra-sons.

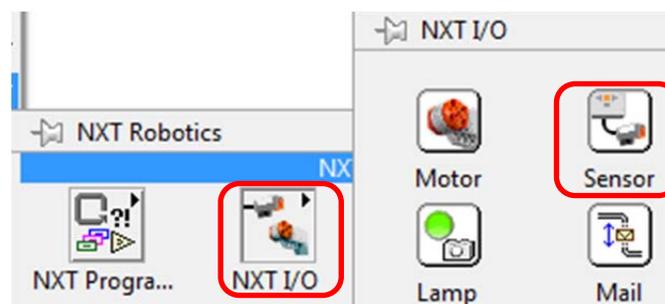
Dans le Robot Project Center, créez un nouveau VI for Computer Target :



Remarque :

- Computer Target signifie : fonctionnement avec câble USB, supervision sur la face avant du VI.
- NXT Target signifie : fonctionnement en mode autonome sans câble USB.

Dans ce nouveau VI, placez un capteur générique :





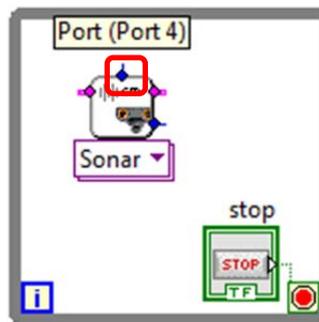
# Sciences et technologies de l'Industrie et du développement durable

Au moyen d'un clic gauche sur le type de capteur, sélectionnez *Read Ultrasonic*. La forme de la fonction s'adapte :

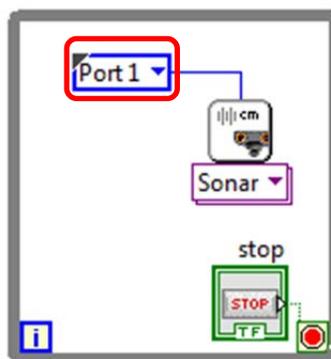


Placez ce capteur dans une boucle While, puis :

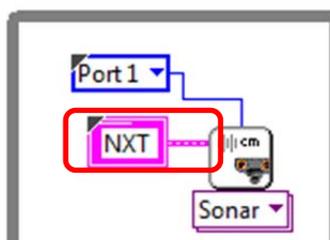
- passez la souris en haut du bloc Sonar jusqu'à ce que le port bleu s'affiche :



- à ce moment, au moyen d'un clic droit, sélectionnez Créer Constante, puis réglez le port conformément à votre câblage : (dans mon cas, le capteur est branché sur le port 1)



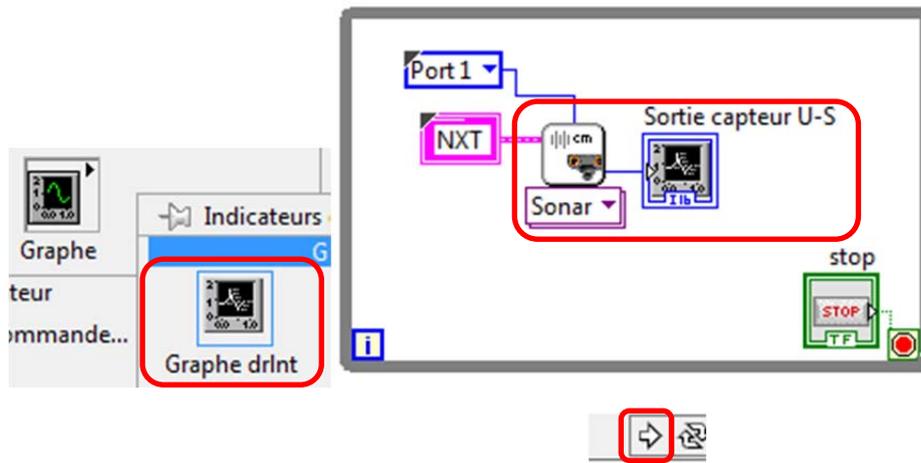
- de même, déclarez la brique utilisée :



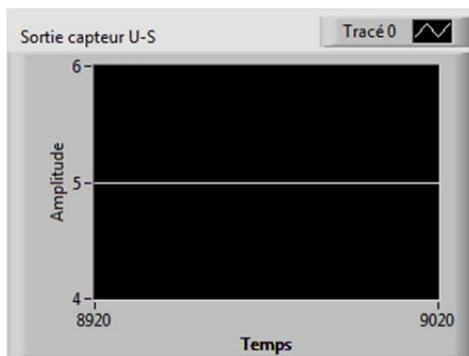


# Sciences et technologies de l'Industrie et du développement durable

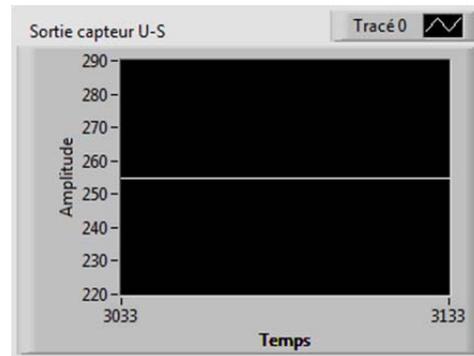
- enfin, placez un graphe déroulant sur la **face avant**, puis câblez-le sur votre **diagramme** :



- lancez le programme en cliquant sur l'icône *Exécuter*  **Exécuter** . Le NXT passe en mode Run :



Cas distance  $\leq 5$  cm

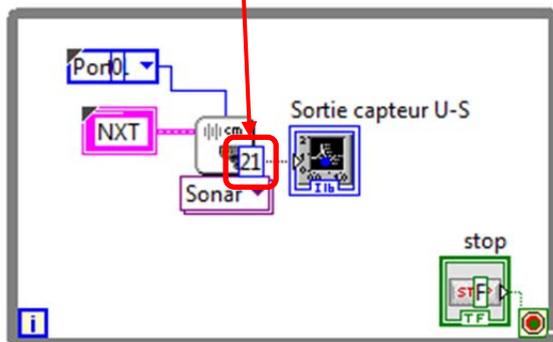


Cas distance  $\infty$

On remarque que le capteur ne peut pas détecter de distance inférieure à 5 cm ni de de distance supérieure à 255 cm. Dans un premier temps, et puisque notre application n'est pas exigeante, nous admettrons que le capteur donne une indication fiable.



On aurait aussi pu monitorer la réponse du capteur en cliquant sur l'icône *Animer l'exécution*. Les valeurs des E/S des fonctions sont alors affichées.





# Sciences et technologies de l'Industrie et du développement durable

## 3.4 Ecrire le programme.

Puisque :

- la distance à tenir est de 5
- la distance maximale mesurée est de 255
- la commande maximale à envoyer au moteur est égale à 100

### 3.4.1 Cas où le robot doit avancer.

Dans le cas où la distance mesurée est supérieure à 5, on peut envoyer une commande C au moteur égale à :

$$C = \frac{100}{255-5} \cdot (\text{Distance} - 5), \text{ soit}$$

$$C = 0,4 \cdot (\text{Distance} - 5)$$

Dans le cas où la distance mesurée est de

- 5 cm, la puissance envoyée au moteur est nulle ;
- 255 cm ou plus, la puissance envoyée au moteur est égale à 100.

### 3.4.2 Cas où le robot doit reculer.

Il serait tentant de vouloir envoyer une commande négative au moteur. Mais ce n'est pas possible. Le facteur de commande allant de 0 à 100 est uniquement représentatif du rapport cyclique.

Il faudra donc demander au robot de reculer avec la puissance maximale.

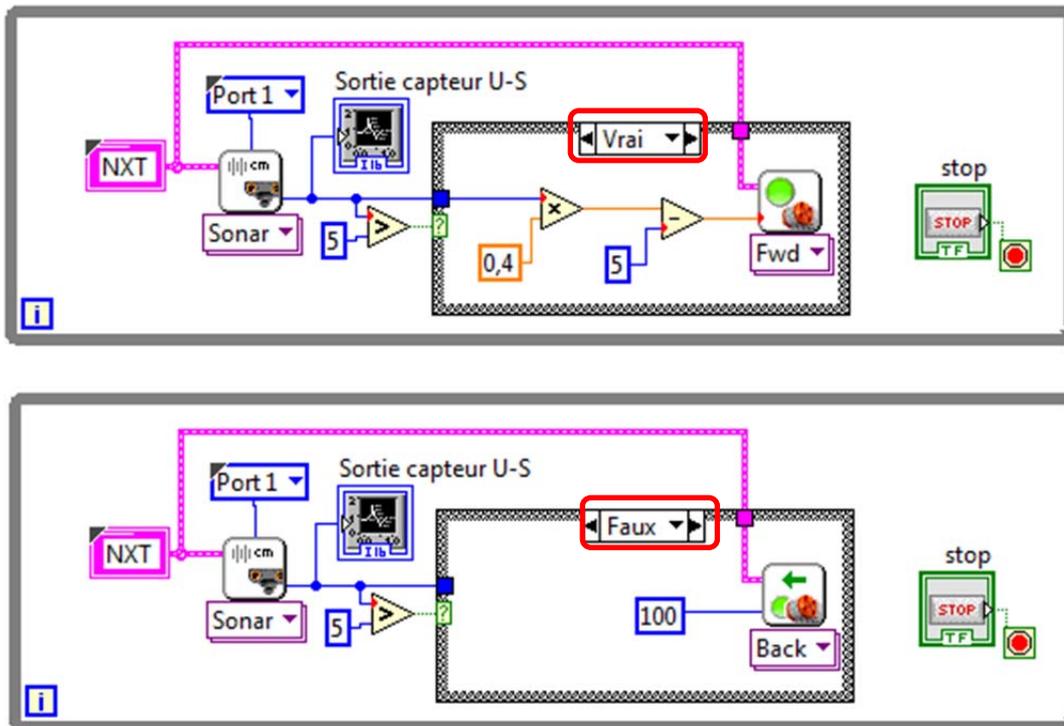
### 3.4.3 Ecriture du programme.

Vous serez donc obligés d'utiliser la structure Case comme indiqué ci-après, et dont votre œil déjà aguerri de programmeur aura compris le fonctionnement suivant :

- Dans le cas où la distance mesurée est strictement supérieure à 5, envoyer une commande égale à  $0,4 \cdot (D-5)$  au moteur, programmé pour avancer (*Fwd*)
- Dans le cas où la distance mesurée n'est pas strictement supérieure à 5, reculer à pleine puissance.



# Sciences et technologies de l'Industrie et du développement durable



Essayez ce programme. Il s'avère que :

- la marche arrière fonctionne bien ;
- la marche avant ne fonctionne pas bien quand le robot est prêt de la main, car la puissance de commande au moteur n'est pas suffisante.

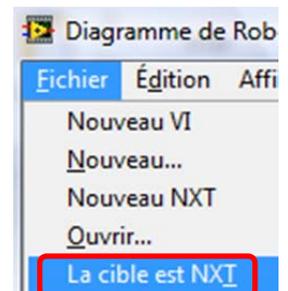
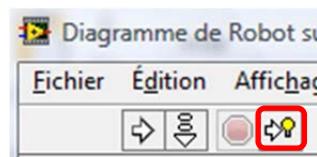
Modifiez le programme de façon à obtenir un meilleur fonctionnement ;

### 3.5 Fonctionnement en mode autonome.

Une fois que votre programme est mis au point, votre robot a pris assez d'assurance pour couper le cordon ombilical !

Passez en mode Cible NXT au moyen de *Fichier / La cible est NXT*.

Pour vérifier que le téléchargement fonctionne bien, vous pouvez faire une Mise au point à l'aide de l'icône associée. Remarquez que la face avant est désactivée.



Ensuite, cliquez sur l'icône déployer  puis la touche orange de la brique pour lancer le programme. Un appui sur la touche *Cancel* (rectangle gris foncé) arrêtera le programme.

Have fun !



# Sciences et technologies de l'Industrie et du développement durable

## 4 Programme à états non combinatoires.

### 4.1 Cas d'étude.

Même s'il a fallu 12 pages pour arriver au programme simple du robot suiveur, et même s'il a fallu dès le premier TD utiliser la boucle CASE, la programmation état simple.

En effet, il n'existait que deux états, et l'entrée dans ces états était de type combinatoire, comme le montre la table de vérité ci-dessous :

Entrée	Etat
$Distance > 5$	Avance
$\overline{Distance > 5}$	Recule

Comment programmer si l'état dans lequel doit se trouver le système dépend non seulement des entrées, mais aussi de l'histoire du système. Par exemple :

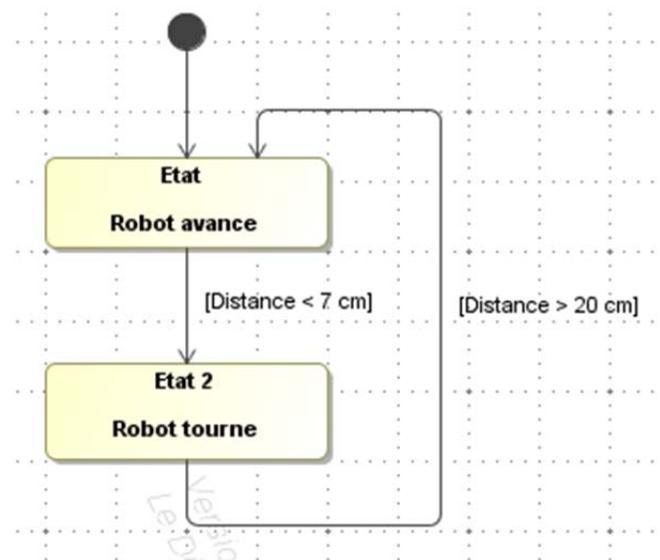
- Etat 1 : Le robot avance jusqu'à se trouver à moins de 10 cm d'un obstacle ;
- Etat 2 : Le robot tourne jusqu'à ce qu'il dispose d'un espace d'au moins 20 cm devant lui.

On voit que pour une entrée de distance égale à 15 cm, le robot peut se trouver soit dans l'état 1, soit dans l'état 2 en fonction de l'histoire du système.

Les machines à état seraient bien utiles, comme le montre le diagramme d'état ci-contre.

Or, à notre grand regret, le NXT n'accepte pas :

- la programmation de bascules ;
- la programmation à l'aide de Statechart.



Il faut donc revenir à nos cours de création de machine à état !.



# Sciences et technologies de l'Industrie et du développement durable

## 4.2 Introduction à la machine d'état.

Soit *Etat n* la variable qui est :

- vraie si l'étape est active ;
- fausse si l'étape n'est pas active.

Pour qu'un *Etat n* soit actif, il faut :

- *Etat n* déjà actif et *Transition n* fausse OU
- *Etat n-1* actif et *Transition n-1* vraie et *Transition n* fausse.

Ce qui donne l'équation classique suivante :

$$Etat_n = \overline{Trans_n} \cdot (Etat_n + Etat_{n-1} \cdot Trans_{n-1})$$

## 4.3 Conséquences pour le cas étudié.

Ceci nous donne les équations suivantes pour notre problème :

- Pour l'état Robot avance :  $Etat\ 1 = \overline{D < 7} \cdot (Etat\ 1 + Etat\ 2 \cdot D < 20)$
- Pour l'état Robot tourne :  $Etat\ 2 = \overline{D > 20} \cdot (Etat\ 2 + Etat\ 1 \cdot D > 7)$

Les actions associées aux états sont :

- Etape 1 : Les deux moteurs avancent, commandés à 100%.
- Etape 2 : Un moteur avance, l'autre recule, tous deux commandés à 100%.

A ce stade, la réflexion préparatoire est intégralement terminée. Il n'y a plus qu'à implémenter ce programme dans Labview.

## 4.4 Implémentation Labview.

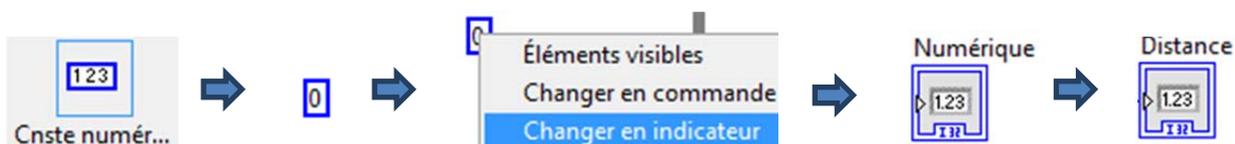
### 4.4.1 Déclaration des variables d'entrée et des variables l'état.

Vous avez besoin de deux types de variables :

- les variables d'entrée (ici, la distance) ;
- les variables d'état, qui vont indiquer quelle étape doit être active
  - Etape 1
  - Etape 2

Dans une boucle while, créez ces deux indicateurs booléens, et un indicateur numérique

Pour créer un indicateur numérique, allez chercher Programmation / Numérique / Constante numérique. Faites un clic droit dessus, puis *Changer en indicateur*. Renommez-le.

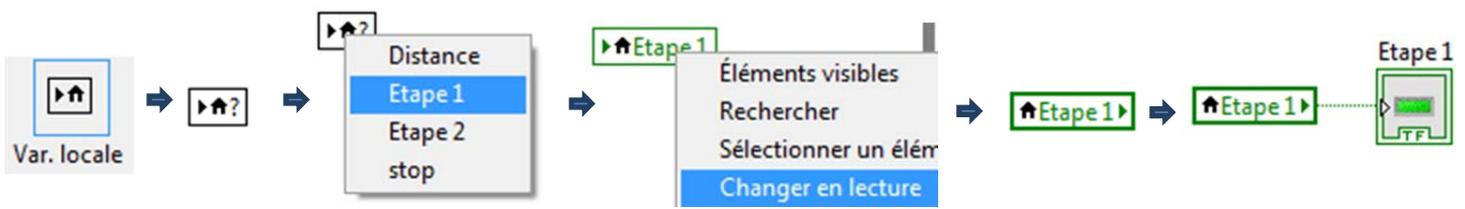
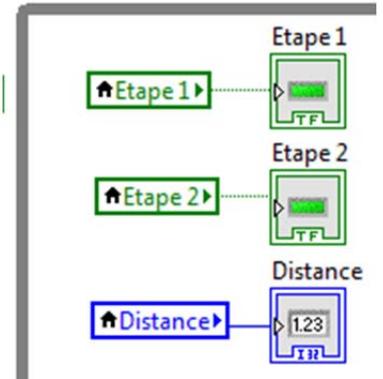




# Sciences et technologies de l'Industrie et du développement durable

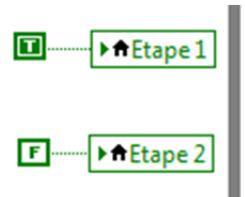
Placez des variables locales, et reliez-les aux indicateurs. Pour cela :

- Allez chercher une variable locale dans *Programmation / Structure / Variable locale* ;
- Déposez cette variable sur le diagramme.
- Choisissez le nom de la variable en faisant un *Clic gauche* dessus, et en choisissant dans la liste.
- Choisissez d'accéder à cette variable en lecture en faisant un clic gauche dessus, puis en choisissant *Changer en lecture*.
- Reliez la variable à l'indicateur.

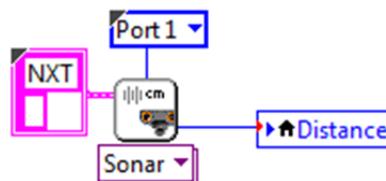


En suivant une procédure analogue, initialisez ces variables à l'extérieur de la boucle While de manière à ce que, à l'initialisation :

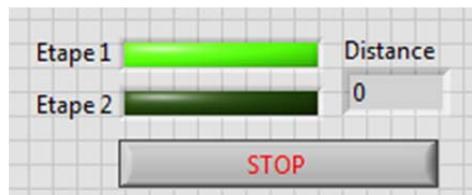
- l'étape 1 soit active ;
- l'étape 2 soit désactivée.



Dans la boucle While, placez le code de lecture de distance :



Organisez la face avant :

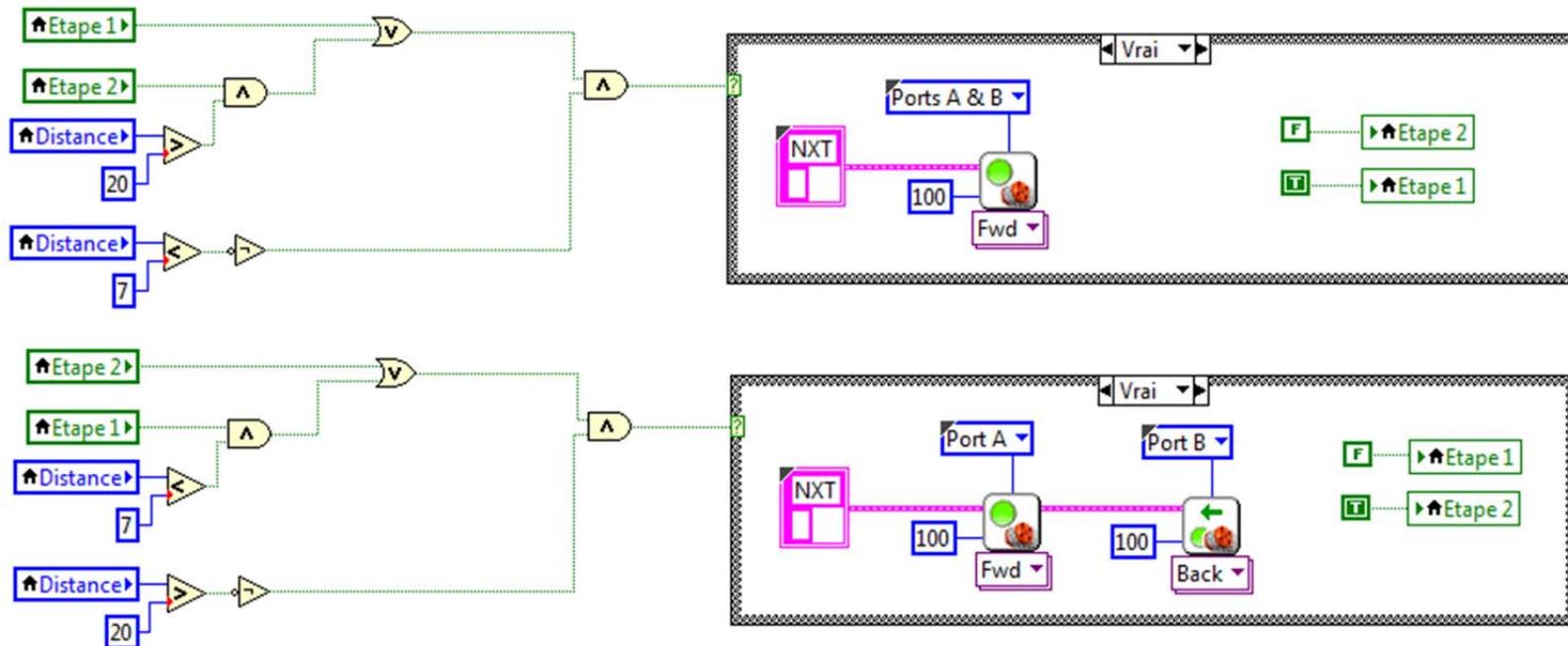


# Sciences et technologies

## de l'Industrie et du développement durable

### 4.4.2 Programmation des états.

Dans la boucle WHILE, placez le code suivant :



Remarques :

- l'étape 1 est l'étape du haut, l'étape 2 celle du bas ;
- les conditions d'activation d'une étape sont les équations que nous avons déterminé au paragraphe 4.3 ;
- il n'y a rien dans le cas *Faux* de chaque boucle Case.
- L'activation / désactivation des Etapes se fait dans la boucle Case de manière à respecter le principe *Traitement des entrées, traitement de l'évolution des états*.



# Sciences et technologies de l'Industrie et du développement durable

## 4.4.3 Test et implémentation.

Testez votre programme et implémentez-le comme indiqué précédemment.

## 5 Evolution.

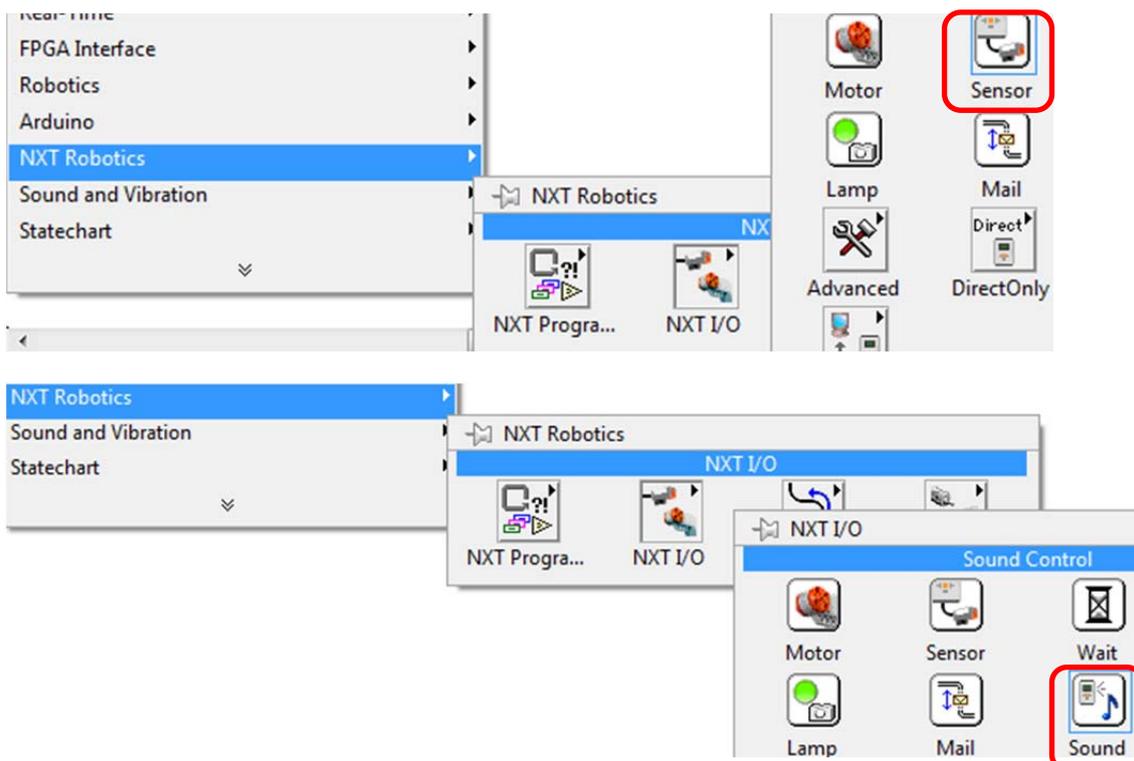
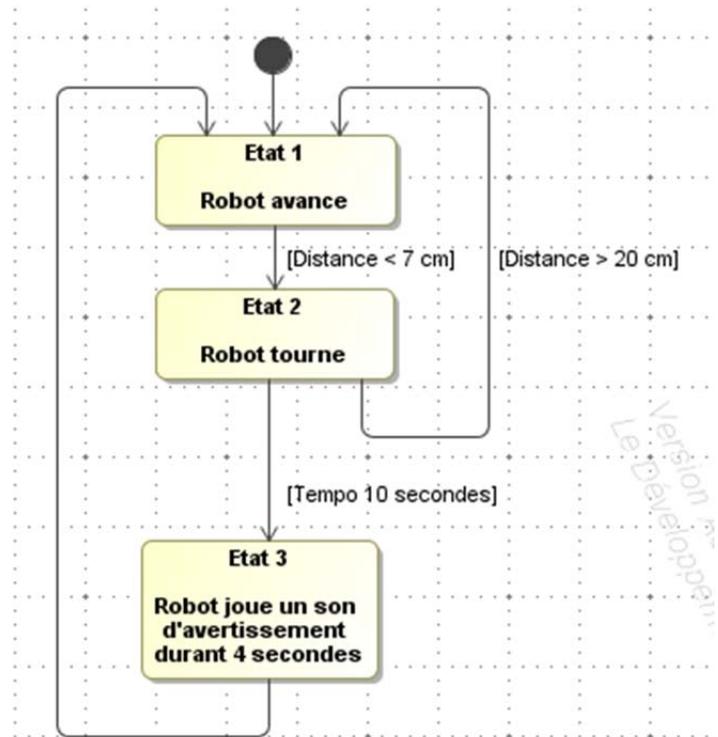
Faites évoluer le programme de manière à ce que, si le robot n'a pas trouvé de solution de chemin au bout de 20 s, le robot s'arrête et joue un son d'avertissement.

Pour cela :

- écrivez les équations d'évolution des états ;
- implémentez votre solution

Le *compteur de temps* est un *Sensor*. Une fois que vous avez placé un *Sensor*, faites un clic droit sur la liste, et choisissez *Read Timer*.

Vous trouverez le son dans le *NXT I/O*, puis *Sound*.





# Sciences et technologies de l'Industrie et du développement durable

## 6 Procédure d'installation

La dernière procédure d'installation à jour est disponible sur le site <http://www.ni.com/academic/mindstorms/>

Deux versions sont disponibles :

- une version Statdalone : un labview indépendant est installé pour les Mindstorms. Cette version est basée sur LV 2010.
- une version Plugin : un plugin se rajoute sur votre Labview.

Nous vous conseillons la version Plugin afin de bénéficier des dernières améliorations de Labview.

