# Projet ACOTRIS

## Réunion de travail

# Objectifs de la réunion

*1) Passerelle UML/Signal*

➔réflexion sur la génération automatique avec langage J

➔programme de travail

➔partage des tâches entre CEA/CS SI

*2) Réflexion sur une Méthode Matlab-Simulink*

*3) Déploiement de l'environnement de simulation UML/Accord*

# Passerelle UML/Signal

# Passerelle UML/Signal

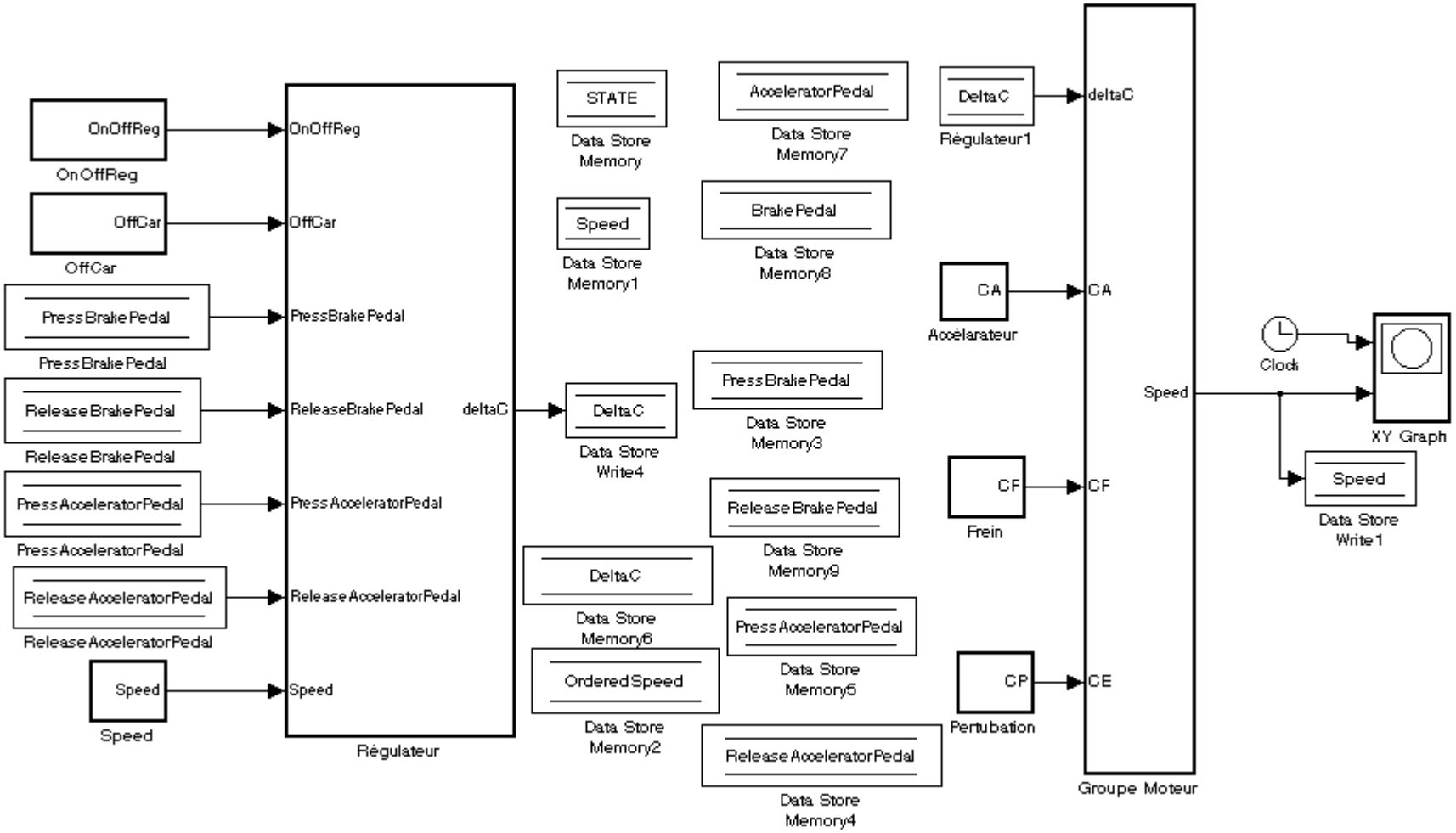
*1) Modèle Matlab-Simulink du Régulateur de vitesse*

*2) Modèle Signal du Régulateur de vitesse*

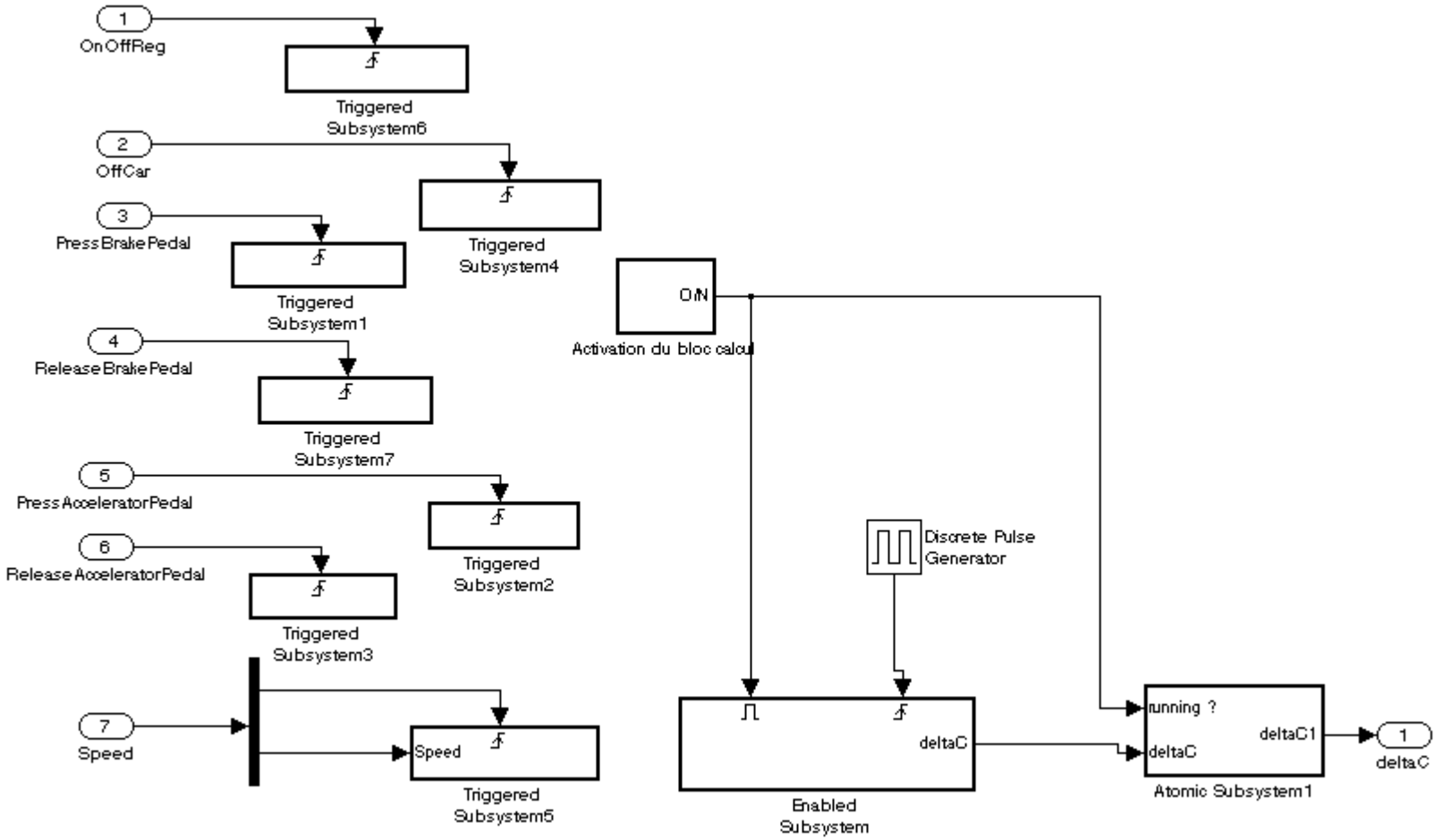
*3) Pistes pour l'établissement de correspondances entre UML/Signal*

# *1) Modèle Matlab-Simulink du Régulateur de vitesse*

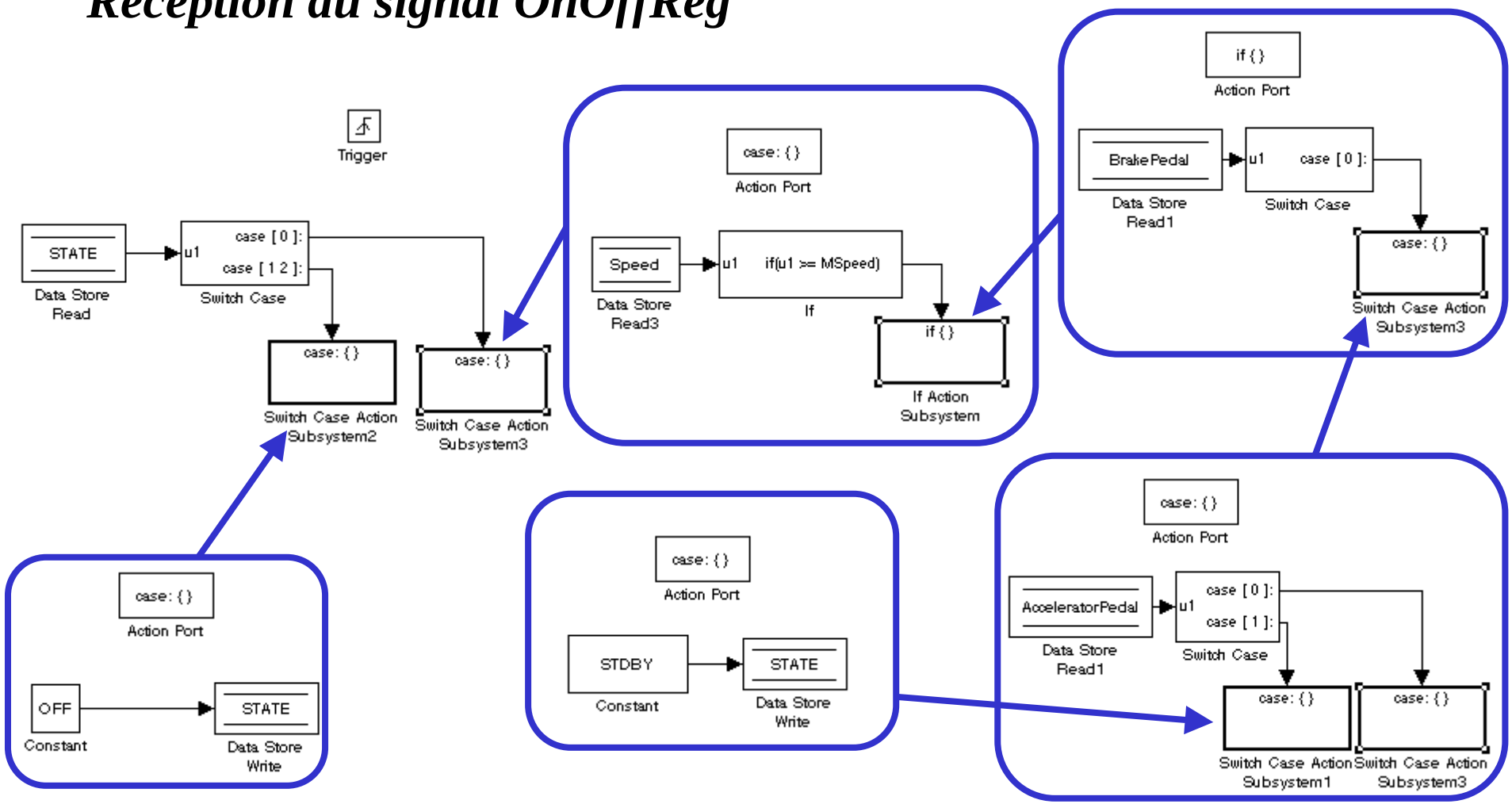
# *Le Régulateur de vitesse et son environnement*



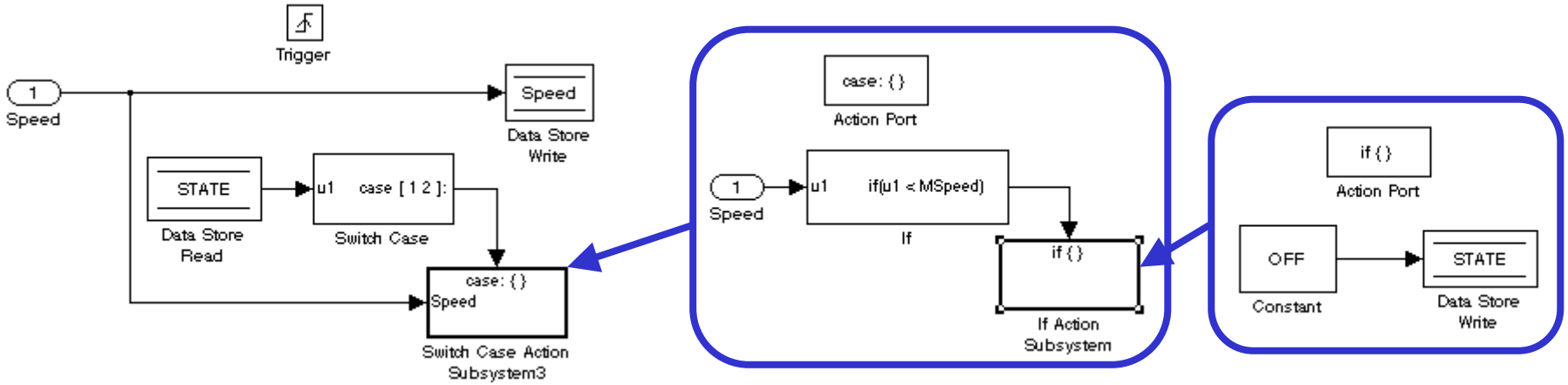
# *Bloc Régulateur de vitesse*



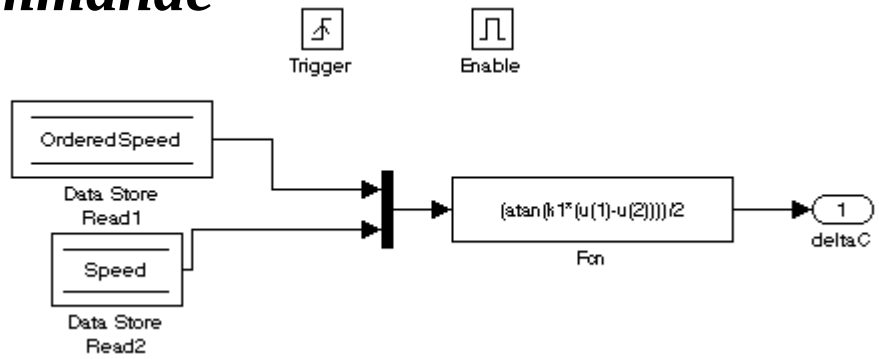
# *Réception du signal OnOffReg*



# *Réception du signal Speed*




# *Calcul de la commande*



# *2) Modèle Signal du Régulateur de vitesse*

## *Structure du programme Signal*

*Choix de modélisation :*

Système à modéliser = Process

*Composition du programme :*

- ➔ déclaration paramètres constants,
- ➔ déclaration des signaux d'entrée/sortie,
- ➔ corps du processus,
- ➔ déclaration des signaux locaux,
- ➔ déclaration/définition des sous-processus.

***Les paramètres constants représentent :***

➔ les états possibles du régulateur de vitesse,
(ex : RUNNING, STDBY, OFF.)

➔ des constantes pour les tests ou les calculs,
(ex : K, MSPEED.)

➔ des valeurs initiales de signaux.
(ex : SPEED0.)

***Les signaux d'entrée peuvent :***

➔ changer l'état du régulateur de vitesse,
(ex : ONOFF_REG, OFFCAR.)

et/ou ➔ déclencher des traitements,
(ex : PRESS_ACC_PED, TOP.)

et/ou ➔ cadencer le signal de sortie.
(ex : TOP.)

***Les signaux locaux représentent :***

➔ l'état du régulateur de vitesse,
(ex : STATE_REG, ZSTATE_REG.)

➔ des grandeurs utiles,
(ex : OSPEED, STATE_BRK_PED.)

➔ des signaux « propagés ».
(ex : STATE_REGcTOP, SPEEDcTOP.)

***Le corps du programme comprend :***

➔ cinq traitements déclenchés par les signaux d'entrée :

- mémorisation de l'état de la pédale de frein,
- mémorisation de l'état de la pédale d'accélération,
- positionnement de la vitesse de consigne,
- calcul et renvoi de la commande,
- changement de l'état du système.

➔ des équations d'horloges,

➔ des définitions de signaux « propagés ».

# *3) Pistes pour l'établissement de correspondances entre UML/Signal*

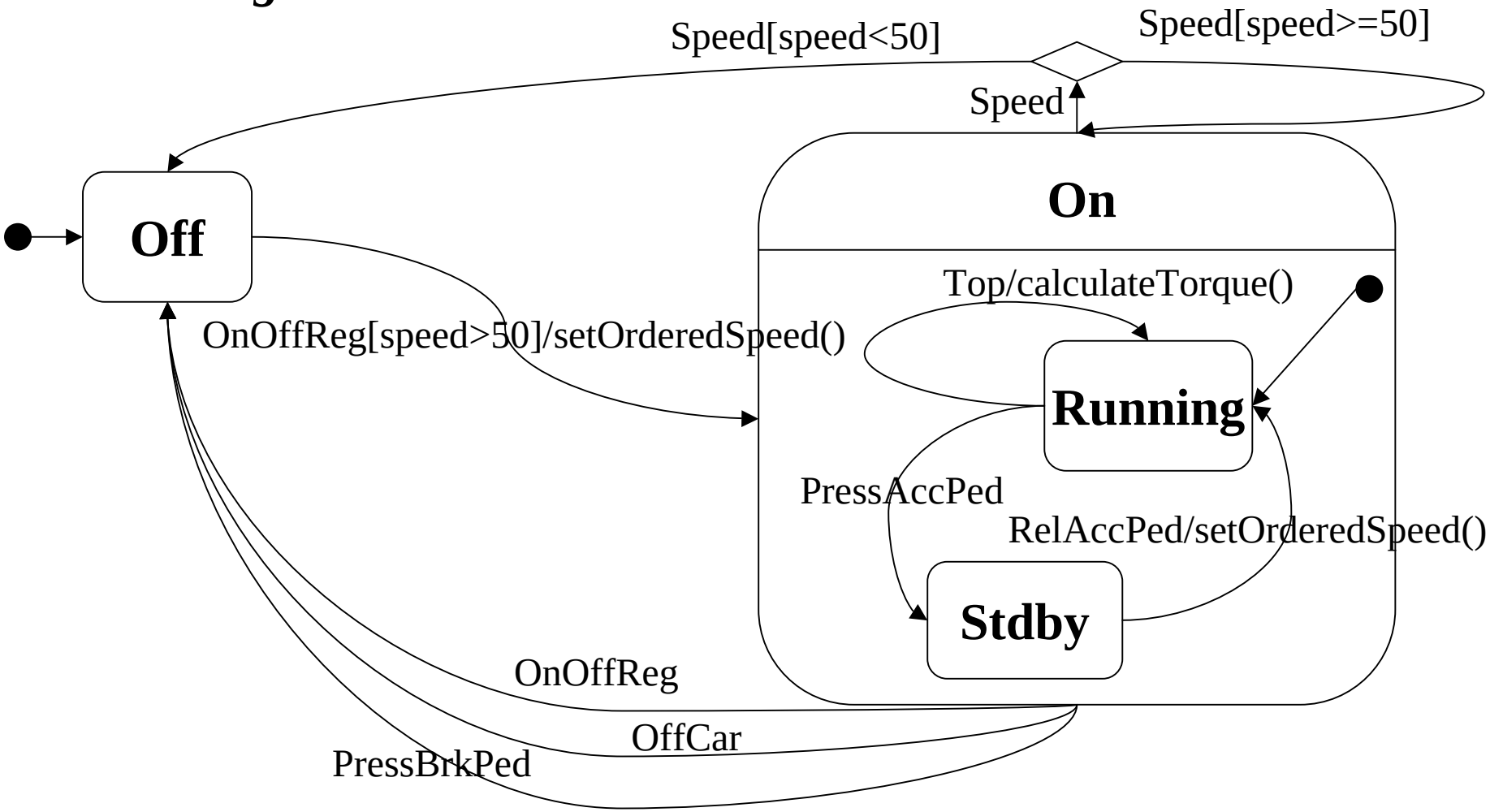
# *Idée de classe UML simple*

| **SpeedRegulator** |
| --- |
| speed<br>orderedSpeed<br>stateBrkPed<br>stateAccPed |
| setOrderedSpeed()<br>pressBrkPed()<br>relBrkPed()<br>pressAccPed()<br>relAccPed()<br>calculateTorque() |

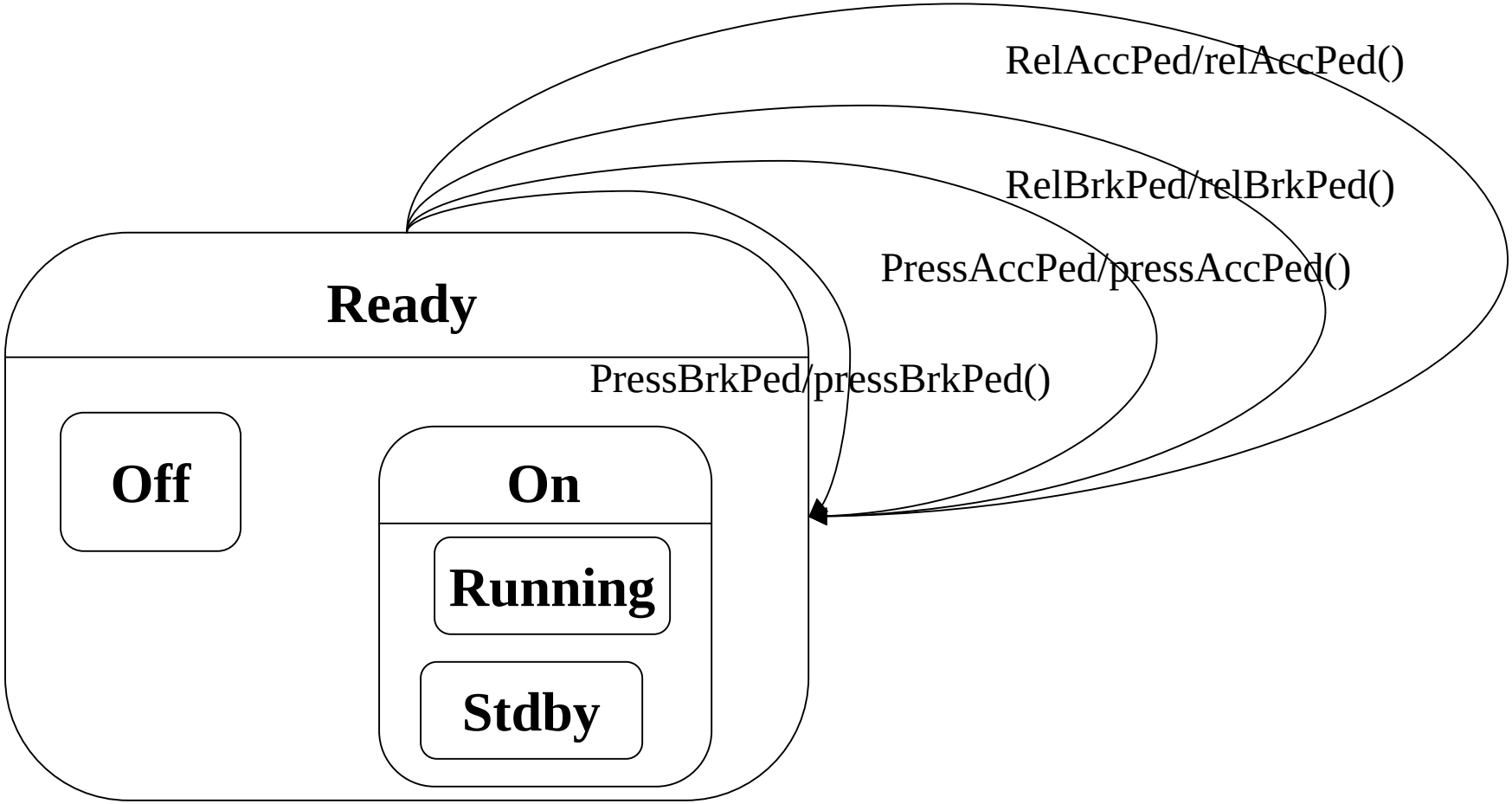
*Une classe sensible aux signaux :*

OnOffReg,
OffCar,
Speed,
PressBrkPed,
PressAccPed,
RelAccPed,
RelBrkPed,
Top ?

# *Idée de diagramme d'états-transitions :*



# *Idée de diagramme d'états-transitions (suite):*



# *Idée de correspondances simples UML/Signal*

| UML | | SIGNAL |
|---|---|---|
| Classe | ⟷ | Processus |
| Méthodes | ⟷ | Sous-processus |
| Attributs | ⟷ | Signaux locaux |
| Signaux | ⟷ | Signaux d'entrée |

## *Idée d'une génération du code Signal à partir des diagrammes d'états-transitions*

Trouver des règles pour spécifier automatiquement en Signal :

➔ l'état du système,
(introduction d'un signal local, cas des sous-états ?)

➔ les changements d'état du système,
(utilisation de l'état retard, des événements ?)

➔ les traitements associés aux réceptions d'événements.
(traitements dépendants de l'état ou non)