

# Un poil de méthodo

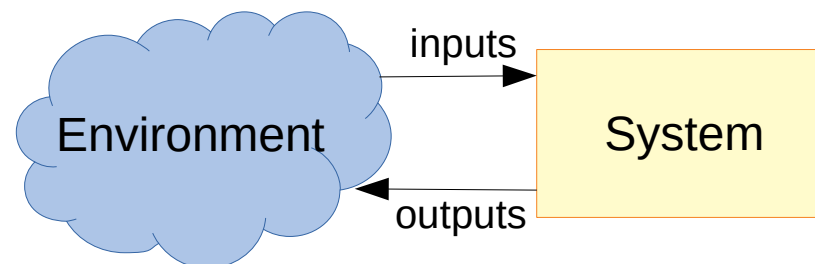
En lien avec le projet...

# Identifier le système et son environnement

- Pour tout ce qui suit, cela peut être fait de différentes manières selon les sensibilités. Ici peu importe nous resterons très informel

# Identifier le système et son environnement

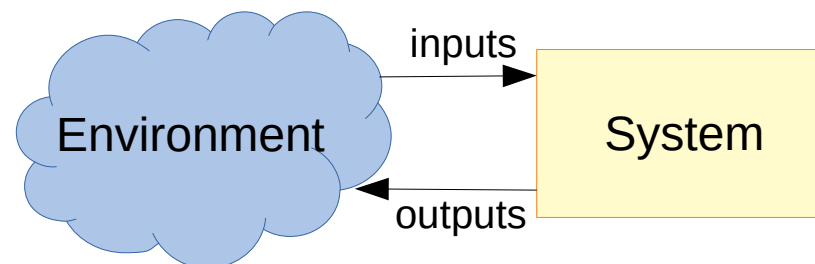
- Pour tout ce qui suit, cela peut être fait de différentes manières selon les sensibilités. Ici peu importe nous resterons très informel
- Il faut identifier ce qui fait partie du système et ce qui fait partie de l'environnement du système.



# Identifier le système et son environnement

- Pour tout ce qui suit, cela peut être fait de différentes manières selon les sensibilités. Ici peu importe nous resterons très informel
- Il faut identifier ce qui fait partie du système et ce qui fait partie de l'environnement du système. Dans le cas de l'exemple du portail,...

We want to model the controller of an entry door by using a FSM.



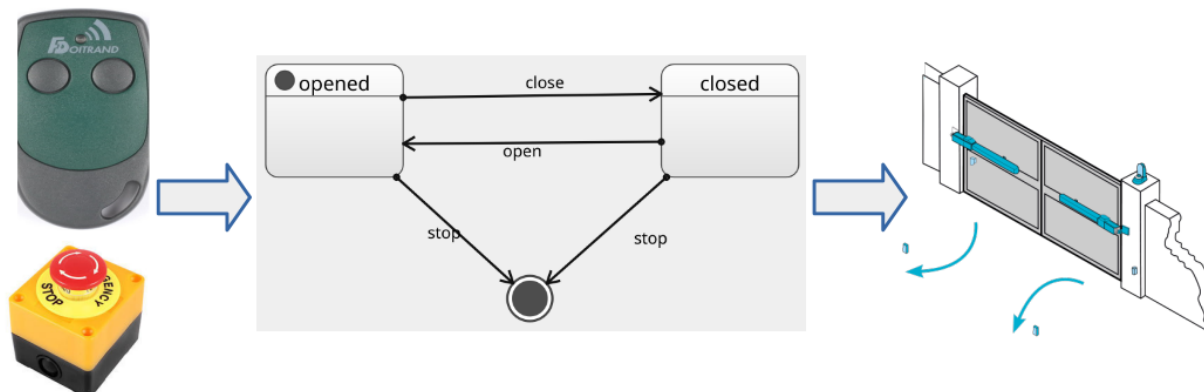
# Identifier le système et son environnement

- Pour tout ce qui suit, cela peut être fait de différentes manières selon les sensibilités. Ici peu importe nous resterons très informel
- Il faut identifier ce qui fait partie du système et ce qui fait partie de l'environnement du système. Dans le cas de l'exemple du portail, la télécommande et le bouton ne font pas partie du système. De plus, seuls les données/événements en relation avec le système sont intéressants.

We want to model the controller of an entry door by using a FSM.

$$\Sigma_I = \{\text{open, close, stop}\}$$

$$\Sigma_O = \{\text{doOpen, doClose}\}$$



# Identifier le système et son environnement

- Pour tout ce qui suit, cela peut être fait de différentes manières selon les sensibilités. Ici peu importe nous resterons très informel
- Il faut identifier ce qui fait partie du système et ce qui fait partie de l'environnement du système. **Si on modélisait le contrôleur de la télécommande ce serait différent...**

We want to model the controller of a remote commande.



# Identifier le système et son environnement

- Pour tout ce qui suit, cela peut être fait de différentes manières selon les sensibilités. Ici peu importe nous resterons très informel
- Il faut identifier ce qui fait partie du système et ce qui fait partie de l'environnement du système. Si on modélisait le contrôleur de la télécommande ce serait différent...

We want to model the controller of a remote commande.

$$\Sigma_I = \{b1, b2\}$$



# Identifier le système et son environnement

- Pour tout ce qui suit, cela peut être fait de différentes manières selon les sensibilités. Ici peu importe nous resterons très informel
- Il faut identifier ce qui fait partie du système et ce qui fait partie de l'environnement du système. Si on modélisait le contrôleur de la télécommande ce serait différents...

We want to model the controller of a remote commande.

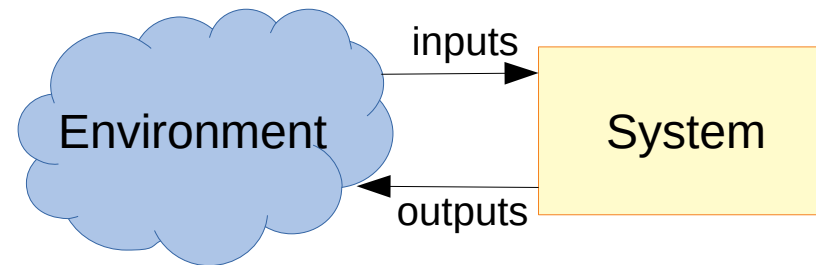
$$\Sigma_I = \{b1, b2\}$$



$$\Sigma_o = \{\text{openLeft}, \text{openAll}, \text{close}\}$$

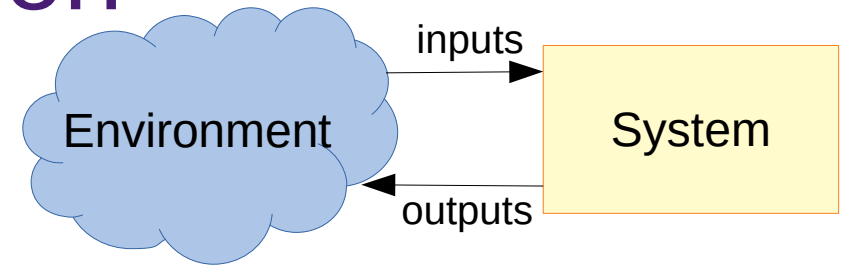


# Identifier le système et son environnement



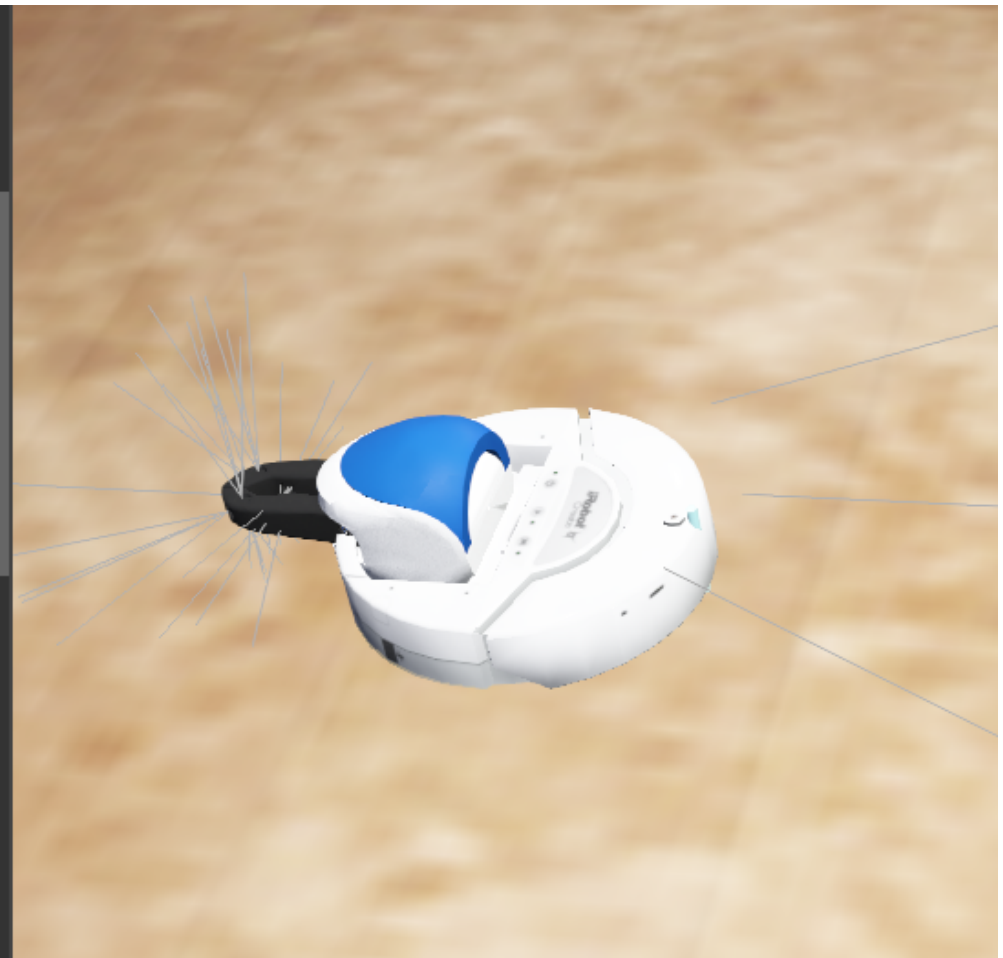
- Dans l'exemple du robot ménager, l'environnement est composé de...

# Identifier le système et son environnement

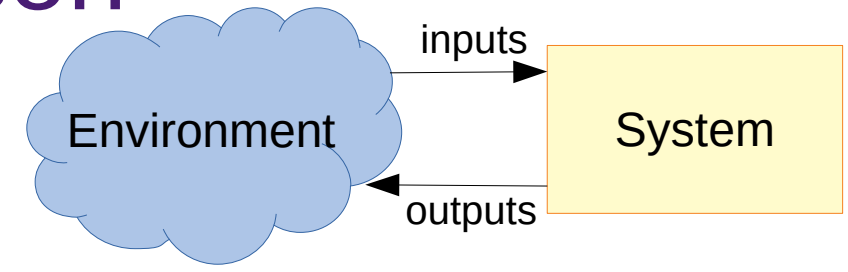


- Dans l'exemple du robot ménager, l'environnement est composé de l'ensemble capteurs/actionneurs liés au robot

```
DEF BODY_SLOT Group
  children
    DistanceSensor "front right distance sensor"
    DistanceSensor "front distance sensor"
    DistanceSensor "front left distance sensor"
    GPS "gps"
    Camera "backCamera"
    Camera "frontCamera"
    DEF GRIPPER Solid
    Pen "pen"
  Shape
  HingeJoint
  HingeJoint
  DEF CREATE_REAR_WHEEL Solid
  DEF CREATE_FRONT_WHEEL Solid
  DEF CREATE_RECEIVER Receiver
  DEF CREATE BUMPER_LEFT TouchSensor
  DEF CREATE BUMPER_RIGHT TouchSensor
  DEF CREATE_LED_ON LED
  DEF CREATE_LED_PLAY LED
  DEF CREATE_LED_STEP LED
  DEF CREATE_CLIFF_SENSOR_RIGHT DistanceSensor
  DEF CREATE_CLIFF_SENSOR_FRONT_RIGHT DistanceSensor
  DEF CREATE_CLIFF_SENSOR_FRONT_LEFT DistanceSensor
  DEF CREATE_CLIFF_SENSOR_LEFT DistanceSensor
```



# Identifier le système et son environnement



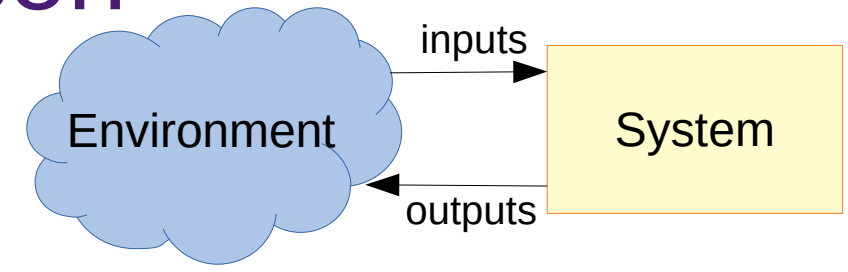
- Dans l'exemple du robot ménager, l'environnement est composé de l'ensemble capteurs/actionneurs liés au robot

```
DEF BODY_SLOT Group
  children
    DistanceSensor "front right distance sensor"
    DistanceSensor "front distance sensor"
    DistanceSensor "front left distance sensor"
    GPS "gps"
    Camera "backCamera"
    Camera "frontCamera"
    DEF GRIPPER Solid
    Pen "pen"
  Shape
  HingeJoint
  HingeJoint
  DEF CREATE_REAR_WHEEL Solid
  DEF CREATE_FRONT_WHEEL Solid
  DEF CREATE_RECEIVER Receiver
  DEF CREATE BUMPER_LEFT TouchSensor
  DEF CREATE BUMPER_RIGHT TouchSensor
  DEF CREATE_LED_ON LED
  DEF CREATE_LED_PLAY LED
  DEF CREATE_LED_STEP LED
  DEF CREATE_CLIFF_SENSOR_RIGHT DistanceSensor
  DEF CREATE_CLIFF_SENSOR_FRONT_RIGHT DistanceSensor
  DEF CREATE_CLIFF_SENSOR_FRONT_LEFT DistanceSensor
  DEF CREATE_CLIFF_SENSOR_LEFT DistanceSensor
```

```
pen: Pen
gripMotors: Motor[]
gripperSensor: DistanceSensor
leftMotor: Motor
rightMotor: Motor
leftSensor: PositionSensor
rightSensor: PositionSensor
ledOn: LED
ledPlay: LED
ledStep: LED
leftBumper: TouchSensor
rightBumper: TouchSensor
leftCliffSensor: DistanceSensor
rightCliffSensor: DistanceSensor
frontLeftCliffSensor: DistanceSensor
frontRightCliffSensor: DistanceSensor
frontDistanceSensor: DistanceSensor
frontLeftDistanceSensor: DistanceSensor
frontRightDistanceSensor: DistanceSensor
frontCamera: Camera
backCamera: Camera
receiver: Receiver
gps: GPS
```

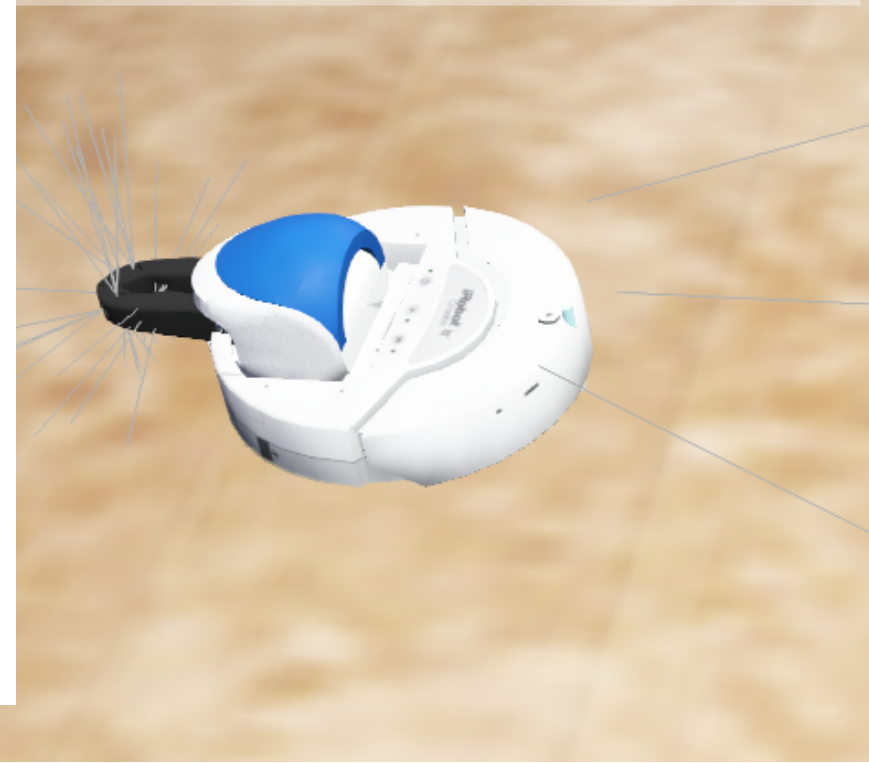


# Identifier le système et son environnement



- Dans l'exemple du robot ménager, l'environnement est composé de l'ensemble capteurs/actionneurs liés au robot

+ exactement c'est l'API permettant d'utiliser les capteurs/Actionneurs qui sera intéressante



```
DEF BODY_SLOT Group
  children
    DistanceSensor "front right distance sensor"
    DistanceSensor "front distance sensor"
    DistanceSensor "front left distance senso"
    GPS "gps"
    Camera "backCamera"
    Camera "frontCamera"
    DEF GRIPPER Solid
    Pen "pen"
  Shape
  HingeJoint
  HingeJoint
  DEF CREATE_REAR_WHEEL Solid
  DEF CREATE_FRONT_WHEEL Solid
  DEF CREATE_RECEIVER Receiver
  DEF CREATE BUMPER_LEFT TouchSensor
  DEF CREATE BUMPER_RIGHT TouchSensor
  DEF CREATE_LED_ON LED
  DEF CREATE_LED_PLAY LED
  DEF CREATE_LED_STEP LED
  DEF CREATE_CLIFF_SENSOR_RIGHT DistanceSensor
  DEF CREATE_CLIFF_SENSOR_FRONT_RIGHT Dist
  DEF CREATE_CLIFF_SENSOR_FRONT_LEFT Distanc
  DEF CREATE_CLIFF_SENSOR_LEFT DistanceSensor
```

```
getPen() : Pen
openGripper() : void
closeGripper() : void
getObjectDistanceToGripper() : double
isThereCollisionAtLeft() : boolean
isThereCollisionAtRight() : boolean
flushIRReceiver() : void
isThereVirtualwall() : boolean
goForward() : void
goBackward() : void
stop() : void
passiveWait(double) : void
randdouble() : double
turn(double) : void
getPosition() : double[]
```

# Plus globalement...

- Vous avez le choix pour cumuler vos différents state chart:
  - un state chart par fonctionnalité gérant les différentes préoccupations ?
  - Un par préoccupation gérant toute les fonctionnalités ?
  - Plein de state charts en réseau ?
  - Un mélange approprié des points précédents ?

# Plus globalement...

- N'hésitez pas à commencer par réfléchir indépendamment de l'outil
- N'hésitez pas à faire de petits tests
  - De petits state charts permettant de tester certaines de vos idées de conceptions.
  - Tester les différentes stratégies précédentes.
  - ...

# Plus globalement...

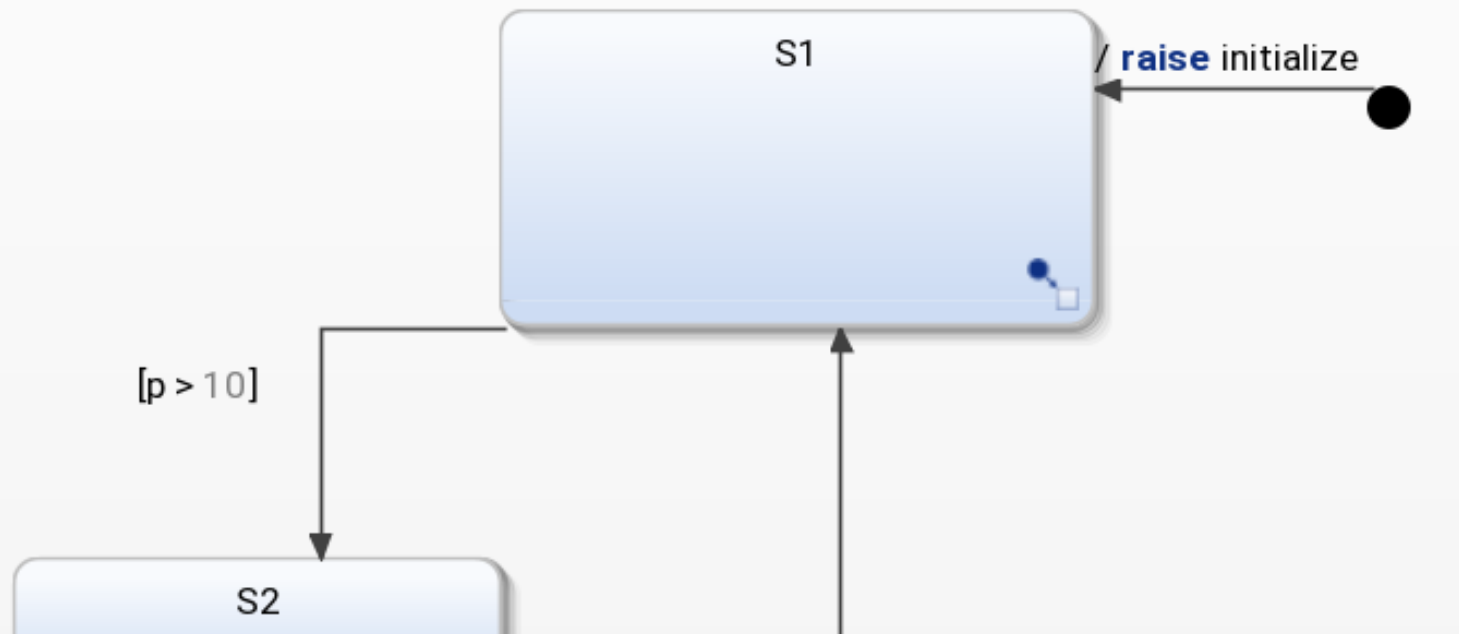
- Dans certaines mesures, il est possible de lire/écrire des données définies dans le state chart depuis le code.
- À priori à éviter mais cela peut s'avérer pratique si l'on veut utiliser des gardes booléennes sur les transitions

```
@EventDriven
// Use the event driven
// Runs a run-to-completion
// each time an event occurs
// Switch to cycle based
// by specifying '@CycleBased'
// instead.
```

```
@ChildFirstExecution
// In composite state
// child states first.
// @ParentFirstExecution
interface:
```

```
var p : integer
```

```
in event bombReached
in event bombGrace
```





# Plus globalement...

- Dans certaines mesures, il est possible de lire/écrire des données définies dans le state chart depuis le code.
- À priori à éviter mais cela peut s'avérer pratique si l'on veut utiliser des gardes booléennes sur les transitions

```
@EventDriven
// Use the event driven
// Runs a run-to-completion
// each time an event occurs
// Switch to cycle based
// by specifying '@CycleBased'
// instead.
```

```
@ChildFirstExecution
// In composite state charts
// child states first.
// @ParentFirstExecution
interface:
```

```
var p : integer
```

```
in event bombReached
in event bombGrace
```

```
private long p;
```

```
public long getP() {
    return p;
}
```

```
public void setP(long value) {
    this.p = value;
}
```

