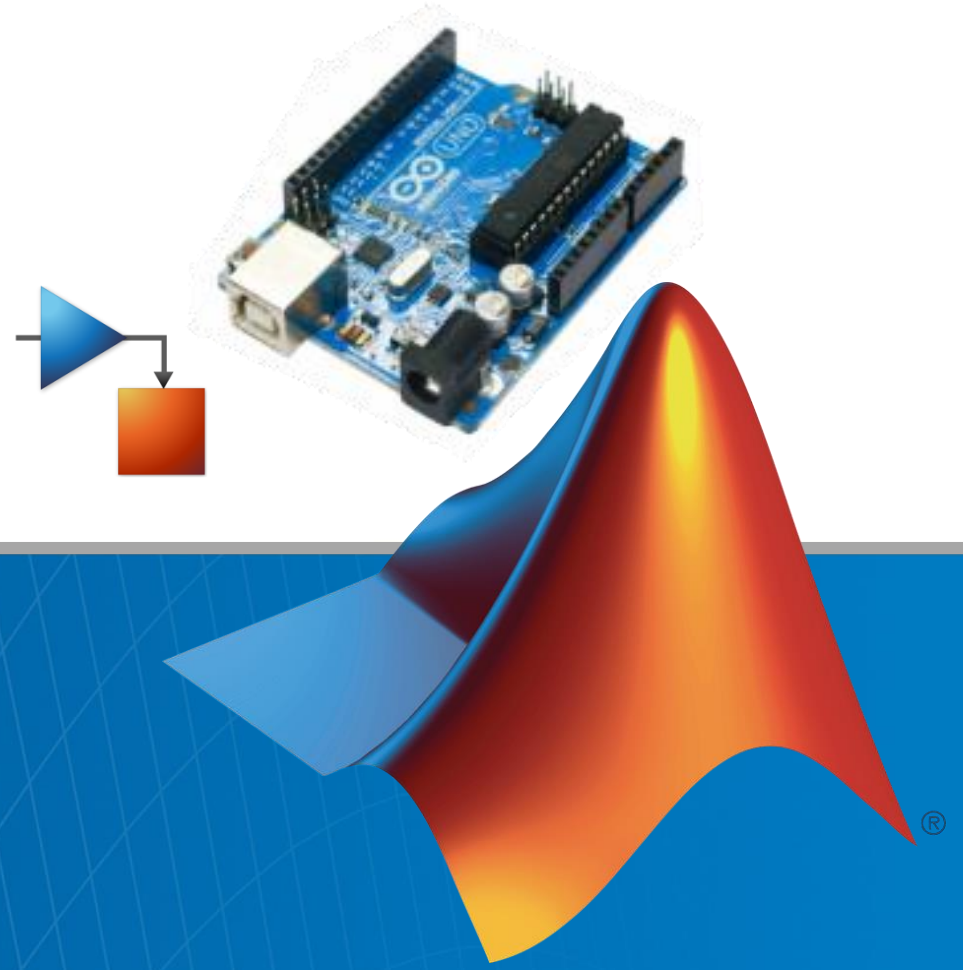
# Workshop Arduino



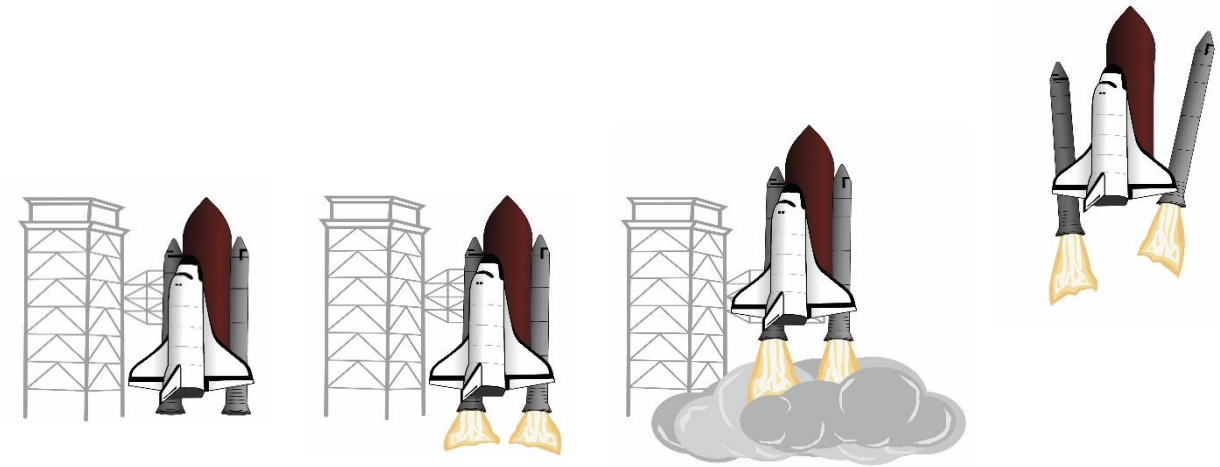
© 2015 The MathWorks, Inc.

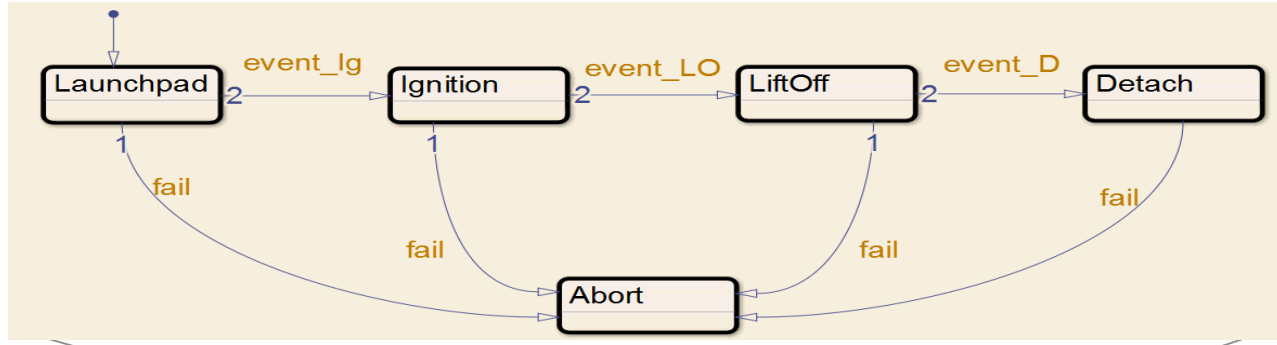
# Démarche de l’ingénieur

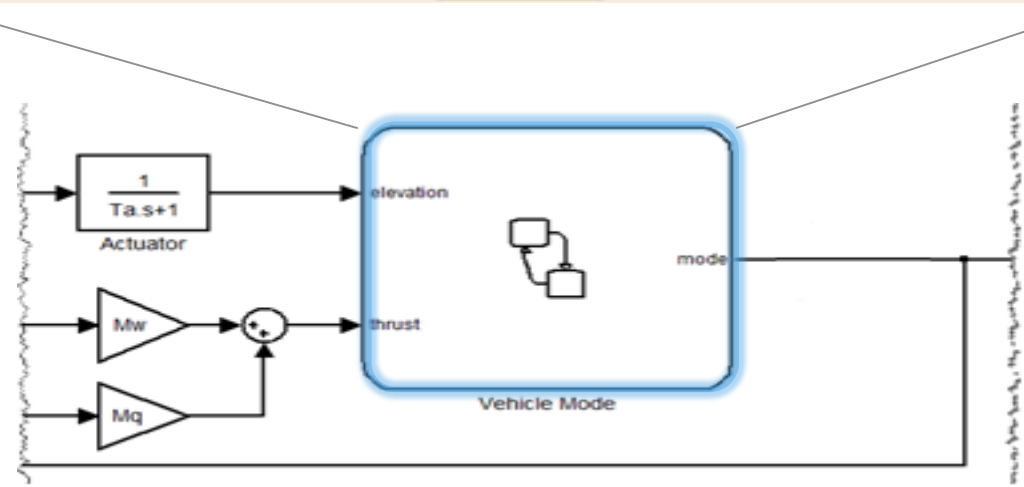
- **Modéliser**
- Simuler
- Implémenter

# Qu’est-ce que Stateflow?

- Modéliser graphiquement et simuler une logique de décision
- Développer la logique à l'aide des machines d'état et des diagrammes de flux
- Voir comment la logique se comporte avec l'animation du diagramme et le débogueur intégré
- Domaines d’application:
  - Logique de supervision
  - Ordonnancement de tâches
  - Gestion d’erreurs

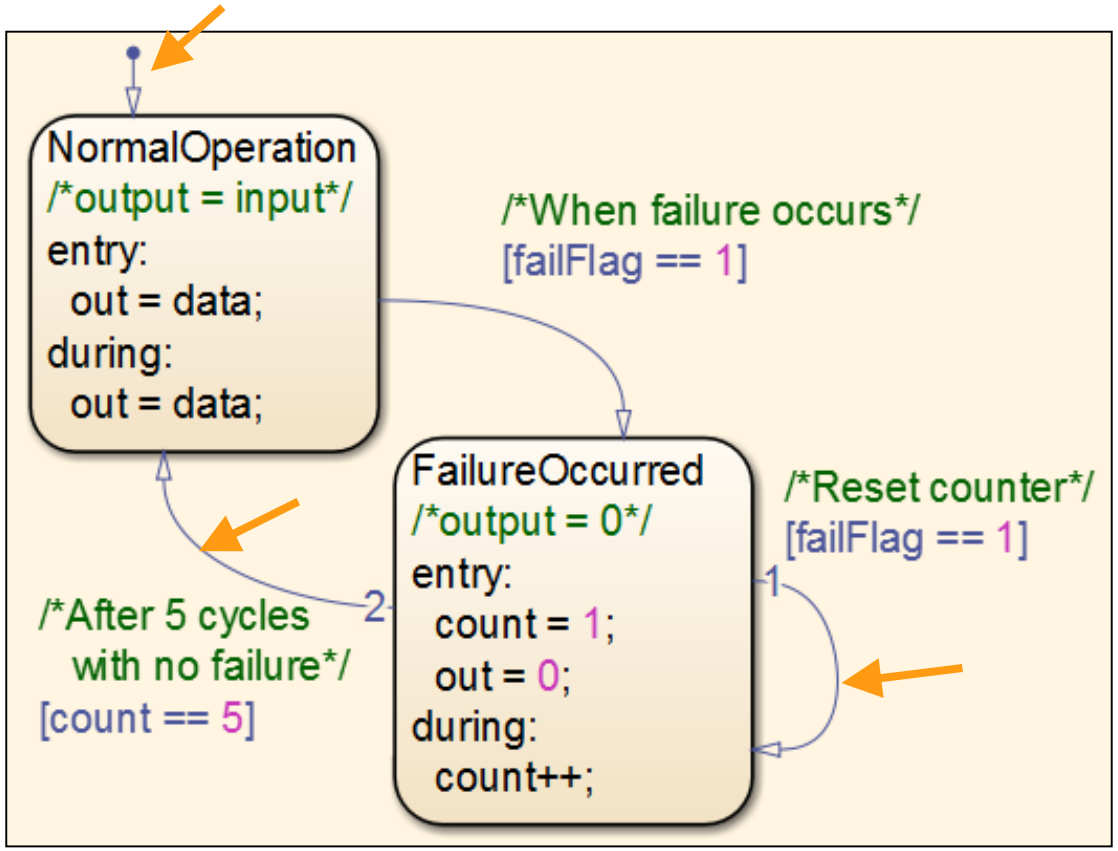




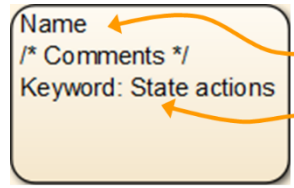



# Syntaxe des étiquettes d'états et de transitions

Transition par défaut




- Etats

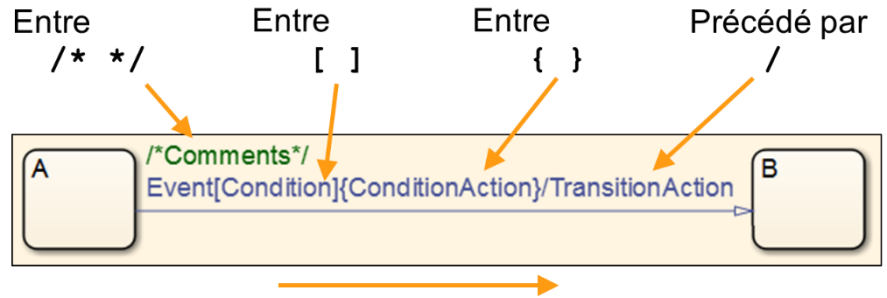



Nom de l'état (obligatoire)

Actions d'état (optionnel)

- **`entry`** – Exécuté lors de l'entrée dans l'état
- **`exit`** – Exécuté lors de la sortie de l'état
- **`during`** – Exécuté lorsque l'état était actif et reste l'état courant (pas de transition)

- Transitions

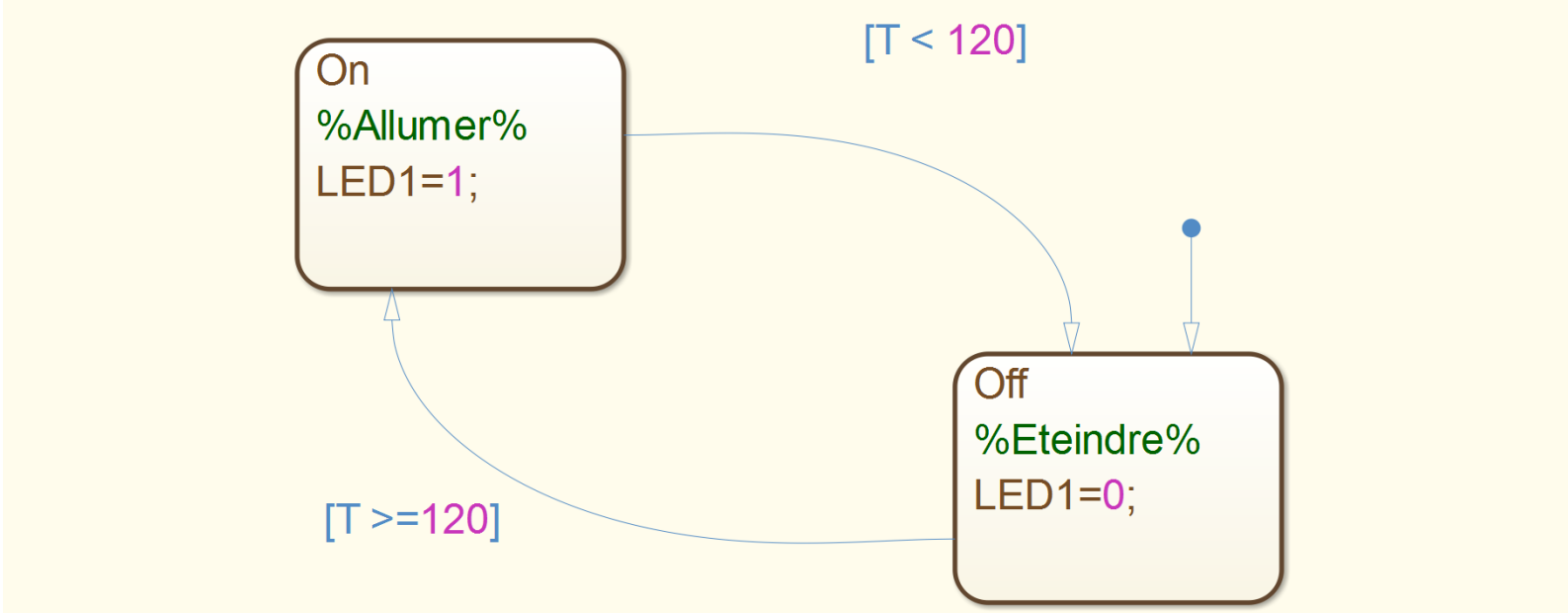



- Les commentaires peuvent être placés n'importe où
- Tous les autres champs doivent apparaitre dans l'ordre

# Exercice 0: Comprendre le Modèle

En se basant sur la syntaxe (page précédente), déduire:

- Les noms des états
- L'état par défaut,
- Les actions prises dans chaque état,
- Les conditions de passage d'un état à un autre

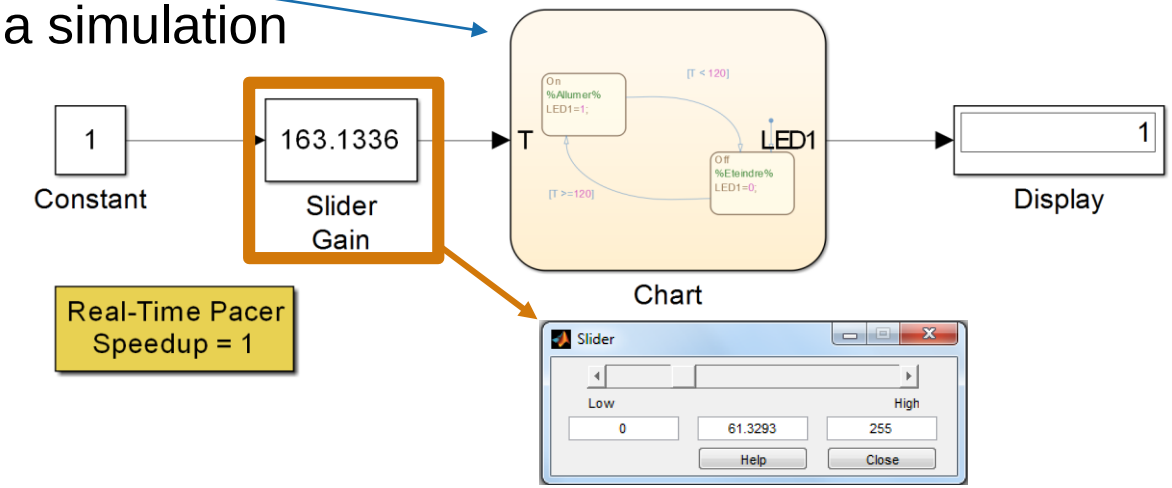


# Démarche de l’ingénieur

- Modéliser
- **Simuler**
- Implémenter

# Exercice 1-a: Simuler le Modèle

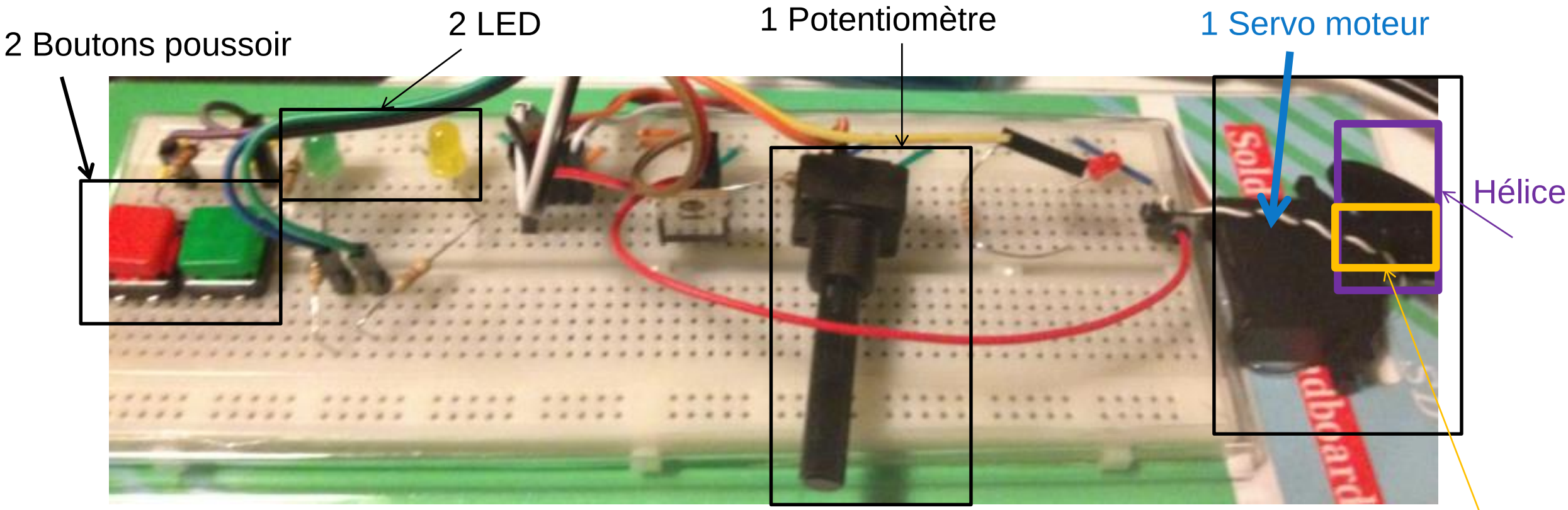
- Ouvrir le modèle Ex1_simulation.slx
- Ajouter le répertoire RealTime_Pacer au path de MATLAB (clic droit, Add to path)
- Simuler le modèle en cliquant sur
  - Double cliquer sur le bloc **Slider Gain**, et à l'aide du curseur, changer les valeurs du signal d'entrée, Qu'observez-vous ?
  - Double-cliquez sur le bloc **Chart** pour accéder au diagramme
  - Cliquer sur pour arrêter la simulation



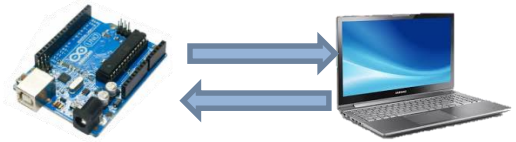
# Démarche de l’ingénieur

- Modéliser
- **Simuler**
- **Implémenter**

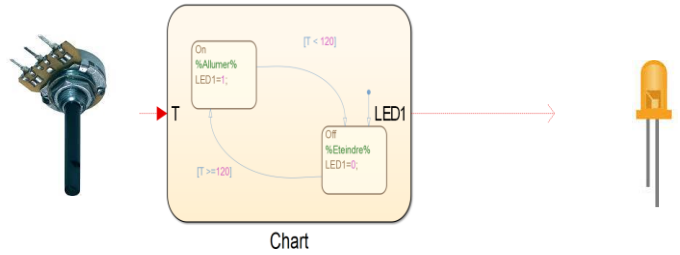
# Présentation de la maquette



# Exercice 1-b: Préparer le modèle pour l'implémenter sur le hardware



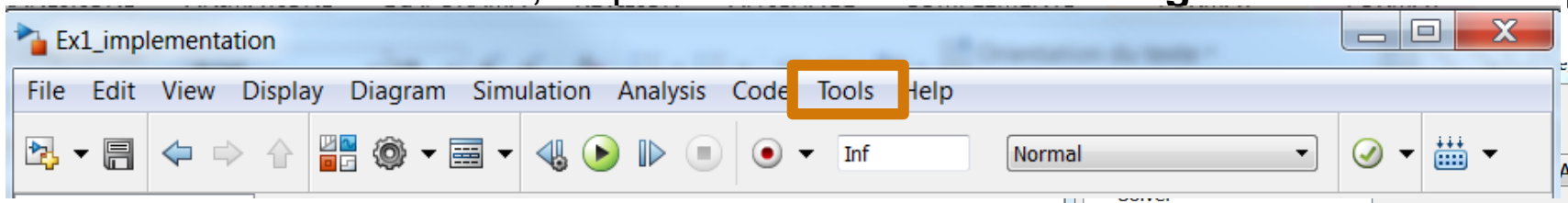
- **Objectif:** Reproduire le comportement du modèle de simulation sur la maquette Arduino en utilisant le potentiomètre pour varier le signal T (température)



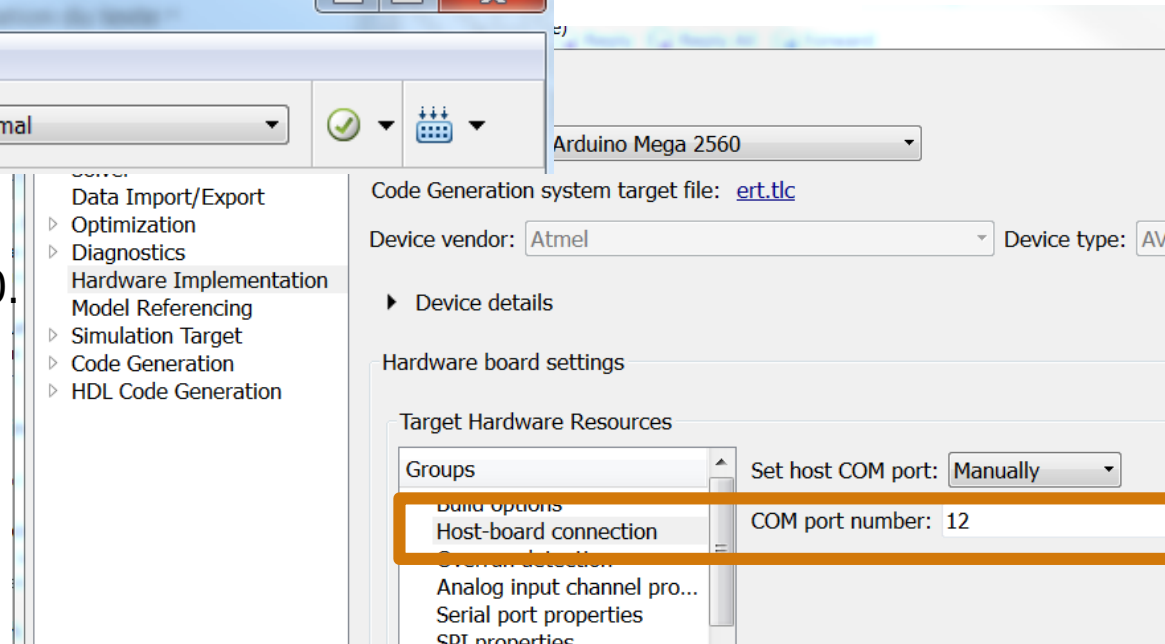

- Ouvrir le modèle Ex1_implementation.slx

Préparer le modèle pour être embarqué sur la cible Arduino tel que:

- Sur la barre d'outil du modèle, cliquer sur **Tools > Run on Target Hardware > Prepare to run…**

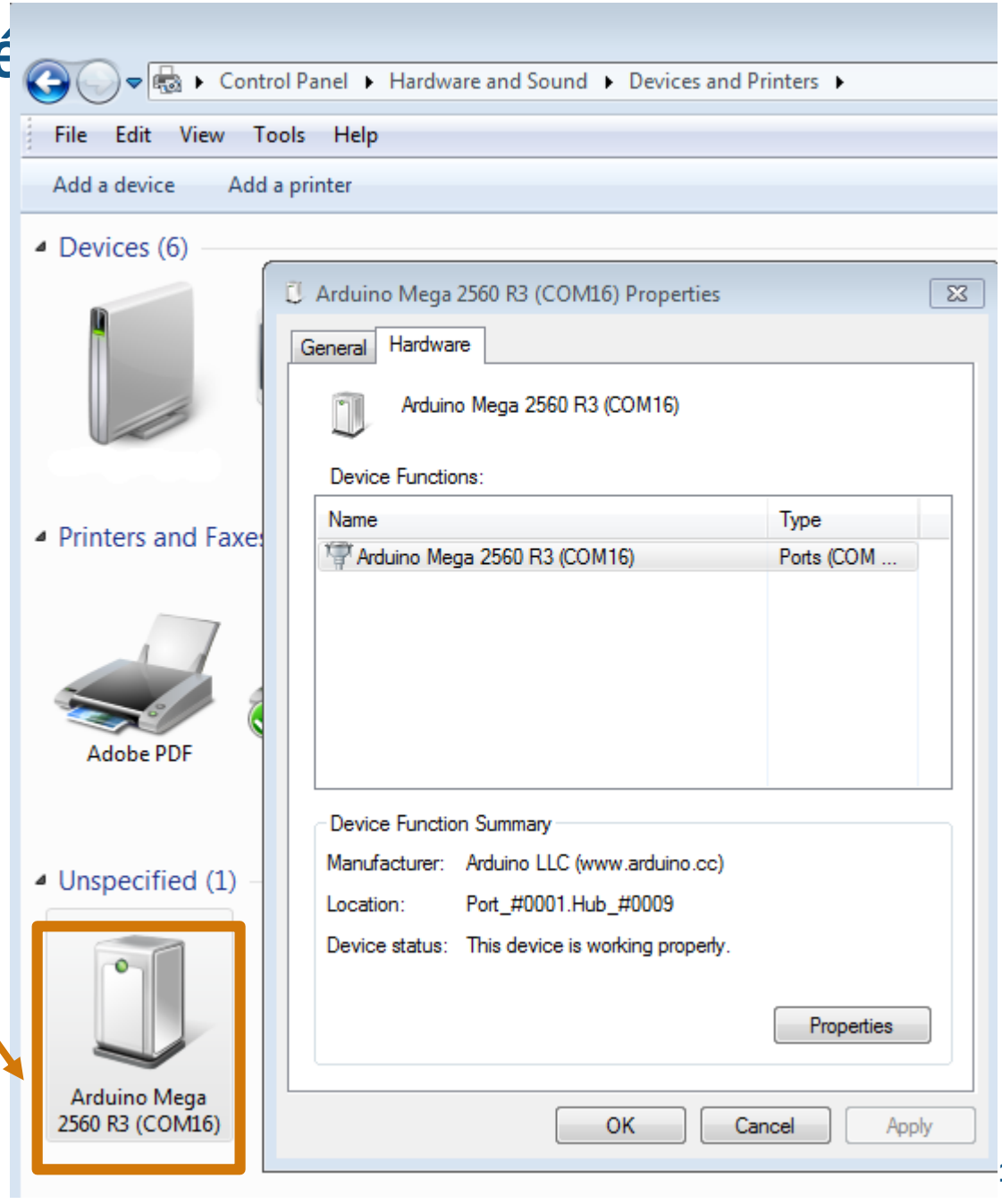



- Attendre que la fenètre suivante s'ouvre
- Choisir la cible (**Target Hardware)** : Arduino Mega 2560.
- Changer Set host COM port à **Manually**
- Renseigner le numéro du port Com(voir Annexe page suivante)



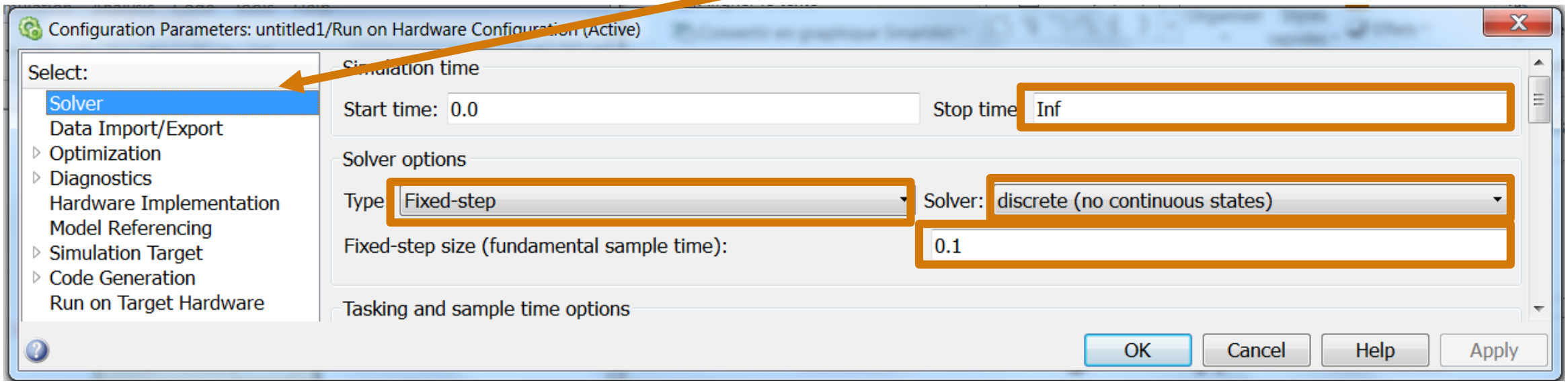
# Annexe: Comment obtenir le numé du port com ?

Aller dans
**Control Panel>Printers and devices.**
Relever le numéro du port Com affiché sur le périphérique **ArduinoMega2560**



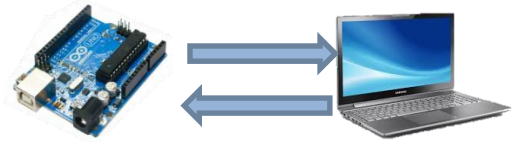
# Exercice 1-b: Préparer le modèle pour l'implémenter sur le hardware

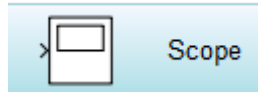
- Avant de cliquer sur OK, aller sur **Solver** et le paramétrer comme suit




- Appuyer sur OK.

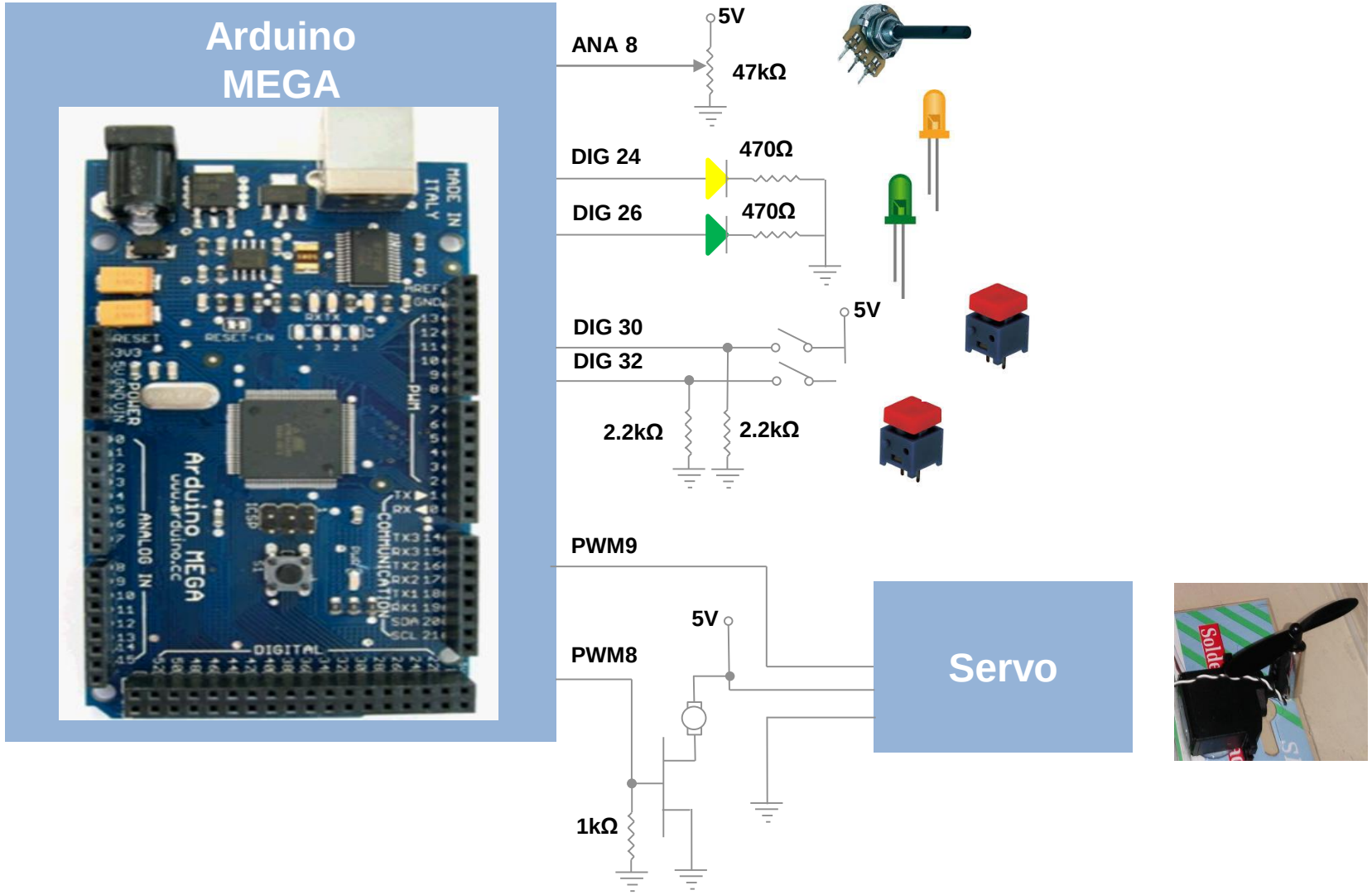
# Exercice 1-b: Compléter le modèle pour l'implémenter sur le hardware



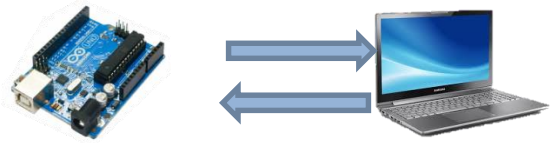
- Compléter le modèle avec les blocs Simulink nécessaires pour l'exécution sur Arduino,
  1. Déterminer le type du signal émis par le potentiomètre et le signal reçu par la LED (Analogique ou Numérique)
  2. Cliquer sur  pour ouvrir la bibliothèque Simulink
  3. Aller dans ***Simulink Support package pour Arduino,*** et glisser les blocs associés aux types de signaux déterminés,
  4. A l'aide du schéma électronique (voir page suivante) ou directement sur la maquette, relever les **pins** sur lesquels sont branchés la LED et le potentiomètre à la carte Arduinio,
  5. Sur le modèle que vous avez complété, configurer les blocs associés à la LED et au potentiomètre avec les numéros des **pins** relevés
  6. Rajouter un bloc Scope* pour visualiser le signal du potentiomètre. 


* Le bloc Scope est dans **« Sinks »** de la bibliothèque Simulink

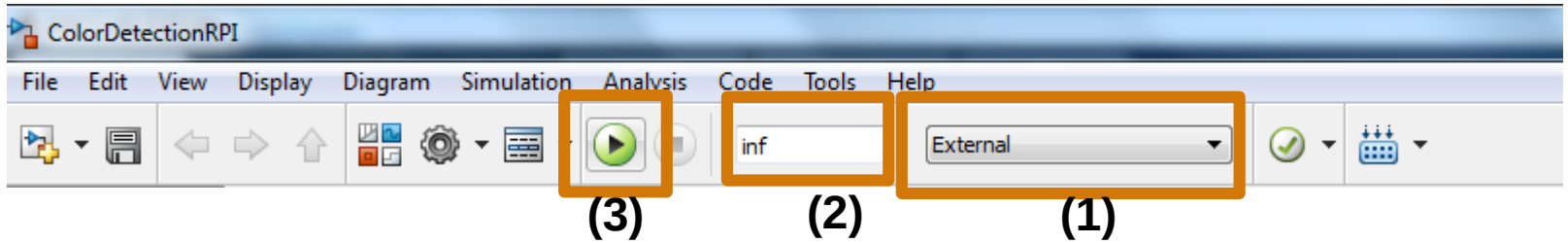
# Schéma électronique



# Exercice 1-b: Exécuter le modèle en mode externe



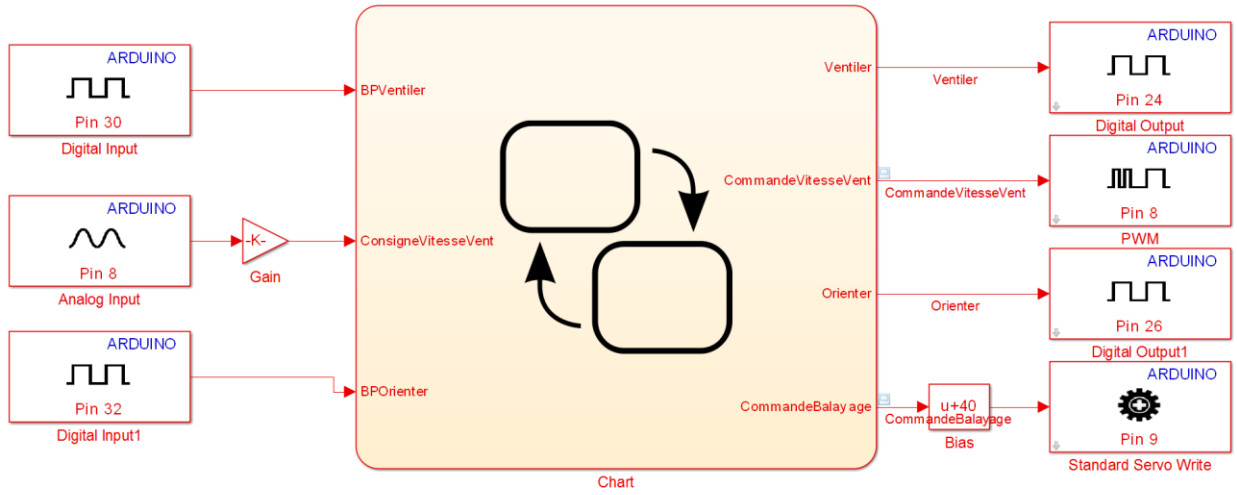
1. Vérifier que la Arduino est allumée et connectée au PC par le cable USB
2. Dans le modèle, sélectionner le mode “**Externe”** (1), verifier que le **temps d’exécution** est à Inf (2) et lancer **l’exécution** du modèle (3)




3. Le code C est généré, compilé, téléchargé et executé sur la carte. La communication entre la carte et le PC permet de modifier les paramètres du modèle et de monitorer les signaux en temps reel pendant l’exécution.
4. Vous pouvez ainsi en live,  tourner le **potentiomètre**, déterminer à l’aide du Scope, l’intervalle de valeurs du signal et observer le passage entre les états On et Off sur le **Chart**, l’allumage et l’extinction de la **LED**.
5. Appuyer sur  pour arrêter le programme

# Exercice 2-a: Comprendre le Modèle du Ventilateur Dyson

- Ouvrir le modèle Dyson_mathworks_mega_external.slx
- Se référer à la page suivante pour apprendre sur le fonctionnement du système **« Ventilateur Dyson »** et comprendre le modèle associé.
- Identifier les **blocs** associés au **potentiomètre** et à la **LED** utilisés dans Exercice 1
- Quel **bloc** représente le **bouton poussoir** qui assure la **ventilation**?
- Identifier sur la maquette le **bouton poussoir** associé et déduire celui qui assure le balayage.
- Quel bloc commande
  - le **moteur** qui assure la **ventilation**
  - le **moteur** qui assure le **balayage**



# Fonctionnement du Ventilateur Dyson

1 Bouton poussoir pour **activer/désactiver** la **ventilation**

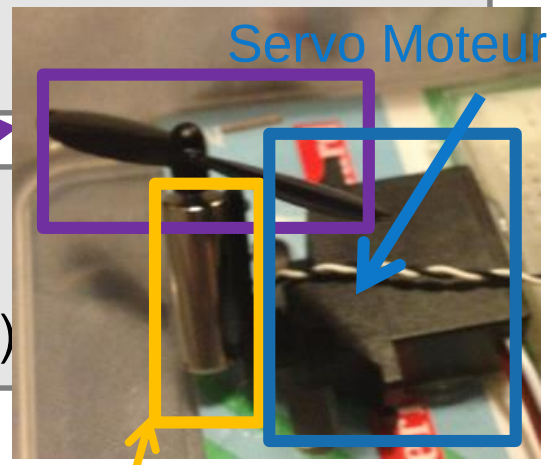
1 Bouton poussoir pour **activer/désactiver le balayage quand il y a ventilation uniquement**

1 LED (Diode) est **allumée** pour indiquer que la **ventilation est ON**

1 LED (Diode) est **allumée** pour indiquer que le **balayage est ON**

**Ventilation ON: le Moteur CC fait tourner l'Hélice sur place**

**Balayage ON: le Servo Moteur tourne l'arbre de l'Hélice** (change l'orientation)




**Potentiomètre** permet de **changer la vitesse de rotation** de l'hélice **(vitesse du Moteur CC)** **quel que soit** l'état du système

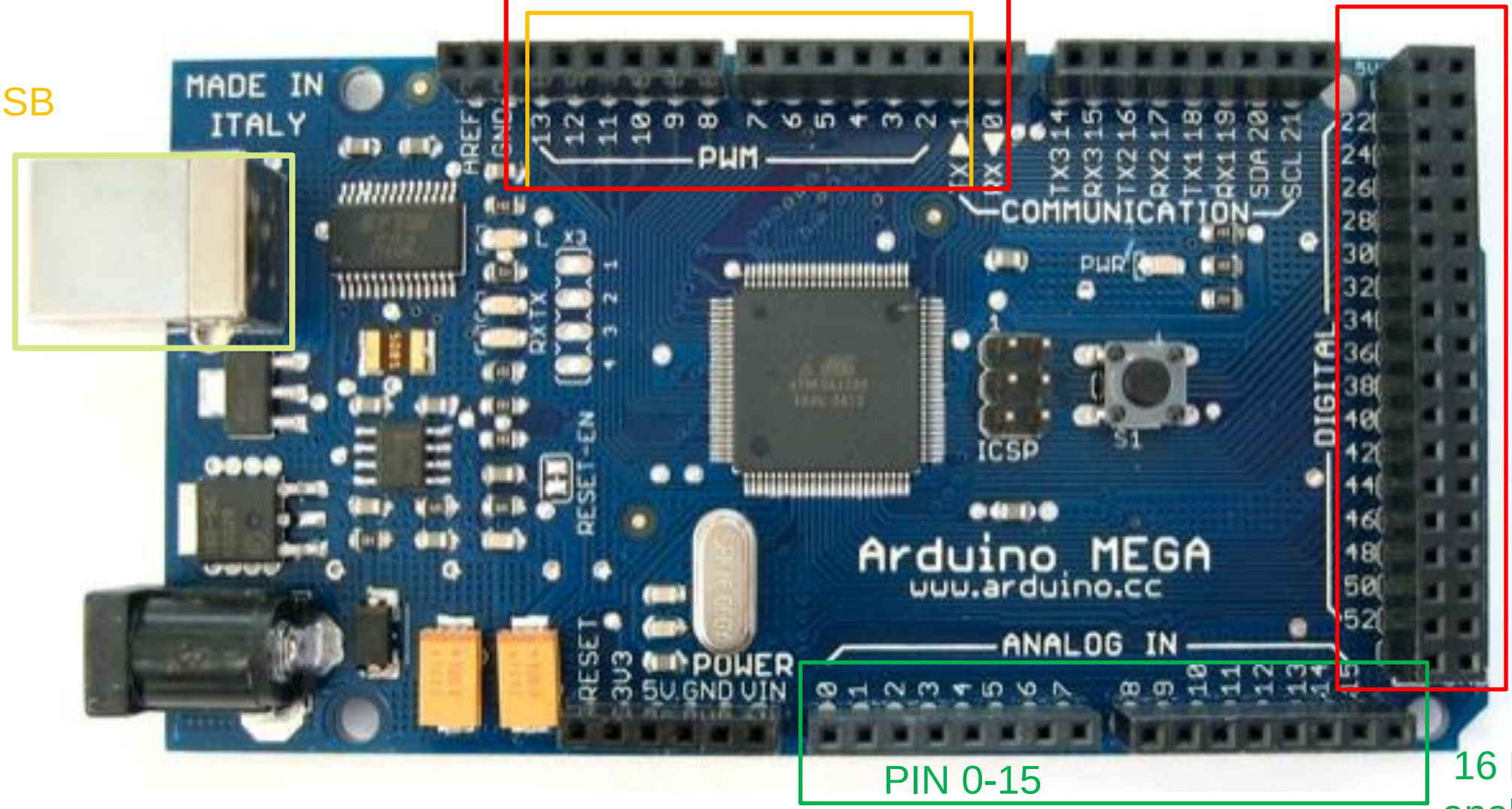
# Exercice 2-b: Implémenter le Modèle du Ventilateur Dyson

- Implémenter le modèle en « **mode externe** » (se référer à l'exercice précédent).
- Une fois le code embarqué,
  - Appuyer sur le **bouton poussoir** pour activer la **ventilation**, et observer l'allumage de la **LED** associée
  - Appuyer sur le **bouton poussoir** pour activer le **balayage** et observer l'allumage de la **LED** associée
  - Faire tourner le **potentiomètre** pour changer la vitesse de ventilation et visualiser à l'aide des scopes du modèle, les signaux de commande envoyés aux moteurs
- Arrêter le programme en appuyant sur 
- Exécuter le modèle en « **mode normal** » en appuyant sur  pour produire un ventilateur autonome (standalone)

# Arduino MEGA

15 sorties PWM PIN 2 à 13 et 44 à 46

USB




54 Entrées /sorties Digitales

PIN 0-15

16 Entrées analogiques

# Coût du prototype

| Description | Prix |
|---|---|
| Arduino  Mega+câble | 22,23 |
| Arduino  Mega boîtier | 5,99 |
| Maquette | 7,5 |
| Servo | 9,9 |
| 47k Potentiomètre | 0,9 |
| 1x Coreless Moteur | 2,09 |
| 1x Hélice | 0,32 |
| 2x Boutons poussoir | 0,9 |
| IRF510 MOSFET | 3,99 |
| 3x LED | 1 |
| 6x Résistances | 2,1 |
| Câbles ruban | 5,99 |
| **Total** | **62,9€** |

# Arduino MEGA details

Summary

| | |
|---|---|
| Operating Voltage | **5V** |
| Input Voltage (recommended) | **7-12V** |
| Input Voltage (limits) | **6-20V** |
| Digital I/O Pins | 54 (of which 15 provide PWM output) |
| Analog Input Pins | 16 |
| DC Current per I/O Pin | 40 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 128 KB of which 4 KB used by bootloader |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Clock Speed | **16 MHz** |