



Le moteur à courant continu

Acquisition des données par Matlab et Arduino

Table des matières

1. Montage expérimental.....	2
2. <i>Modèle Matlab</i>	4
2.1 Le package Matlab pour Arduino.....	4
2.2 Acquisition des données.....	6
2.3 Création du fichier de mesures.....	15
3. <i>Résultats</i>	18

1. Montage expérimental

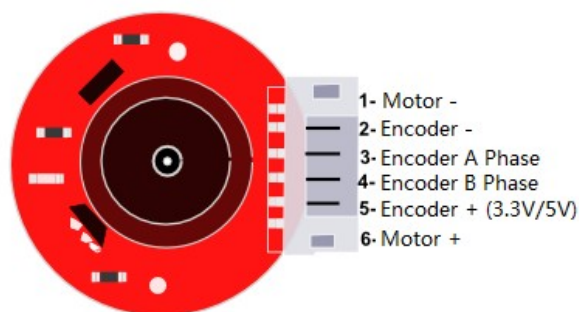
On souhaite mesurer la vitesse de rotation d'un moteur en fonction du temps à partir de l'instant où on applique à ses bornes une tension continue. C'est ce que l'on appelle réponse indicielle du moteur. Dans les activités précédentes on a simulé cette réponse indicielle à l'aide des modèles causal (« Motoreducteur_ModeleCausal.slx ») et acausal (« Motoreducteur_ModeleAcausal.slx »).

Le système simulé était le motoréducteur FIT0520. Ce motoréducteur contient un codeur qui permet de mesurer la vitesse de rotation, simplifiant ainsi le montage expérimental d'acquisition de la réponse indicielle. La procédure d'acquisition et le modèle Matlab correspondant restent les mêmes si l'on utilise n'importe quel autre moteur à courant continu, évidemment avec un codeur externe.

Lors de ces simulations, les paramètres des blocs Matlab ont été soit fournis par la fiche technique du motoréducteur, soit pris égaux à des valeurs standard (par exemple celles des blocs Matlab). Un certain nombre de ces paramètres ont été modifiés afin d'obtenir la réponse indicielle la plus proche de celle expérimentale (fichier « Motoreducteur_mesures.mat »).

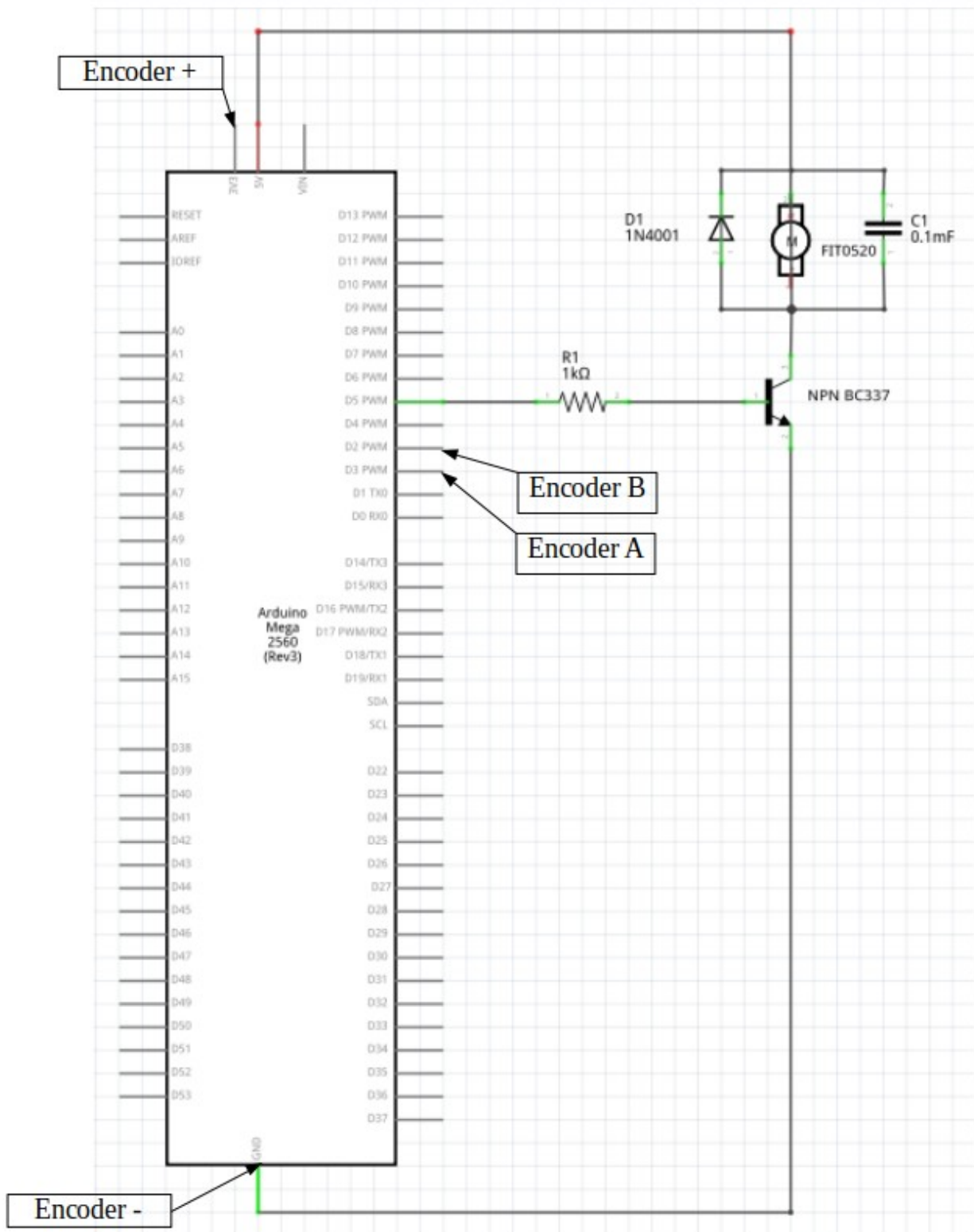
Dans ce TP on va réaliser l'acquisition de la réponse indicielle par Matlab et Arduino du motoréducteur FIT0520 et on va l'écrire dans un fichier de type Matlab. Autrement dit, on va créer le fichier de mesures expérimentales « Motoreducteur_mesures.mat ».

Le motoréducteur FIT0520 a 6 bornes comme montré dans la figure ci-dessous:



Les impulsions du codeur sont comptées sur l'Arduino en utilisant des interruptions. La carte utilisée étant une Arduino Méga les broches permettant l'utilisation des interruptions sont la 2 et la 3. Les bornes « Encoder A » et « Encoder B » du moteur seront donc liées à ces broches.

Le montage utilisé est présenté dans la figure ci-dessous :



Dans le schéma, les composants sont :

- une résistance de $1k\Omega$
- un condensateur de $104\ \mu\text{F}$
- une diode 1N4148
- un transistor NPN BC337
- le motoréducteur FIT0520

La carte Arduino est utilisée aussi pour contrôler la mise en rotation du moteur et, éventuellement sa vitesse. Pour cela, sur le pin 5 est connecté la base d'un transistor NPN (BC337) et se comporte comme un interrupteur qui applique aux bornes de moteur une tension continue de 4,5 V ou 0 V en fonction de la valeur de l'intensité du courant dans sa base. Dans le circuit de la base, la résistance à une valeur de $1\ \text{k}\Omega$.

Lorsque la base du transistor n'est plus alimentée, le moteur ne l'est plus non plus, mais par inertie, il va continuer à tourner un certain temps. Dans ce cas il va se comporter comme un générateur de tension qui risque d'endommager le transistor. Pour empêcher cela on a connecté une diode (« de roue libre », 1N4148) en parallèle avec le moteur .

Afin de filtrer les bruits qui entachent les mesures de vitesse de rotation, on branche un condensateur ($104\ \mu\text{F}$) en parallèle avec le moteur.

2. Modèle Matlab

2.1 Le package Matlab pour Arduino

Le package de prise en charge de MATLAB pour Arduino permet d'écrire des programmes MATLAB qui lisent et écrivent des données sur Arduino et les périphériques connectés.

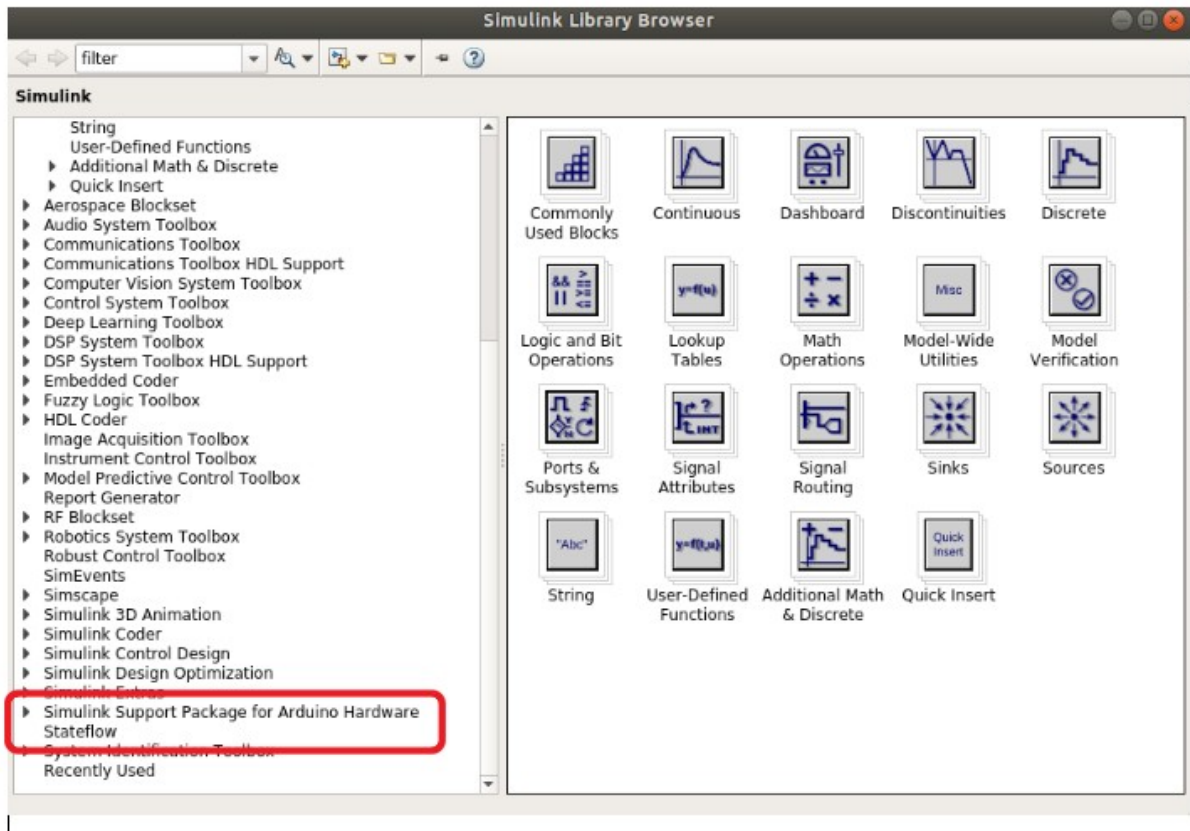
Avec le package de support MATLAB pour Arduino, l'Arduino est connecté à un ordinateur exécutant MATLAB. Une partie du traitement est effectué sur l'ordinateur par Matlab/Simulink et une autre sur l'Arduino. Les deux peuvent communiquer ensemble au travers d'une liaison série (USB).

Cette association peut être utile en S-SI et en STI2D, notamment en phase de projet, en permettant aux élèves de mettre en œuvre une solution de prototypage rapide accessible.



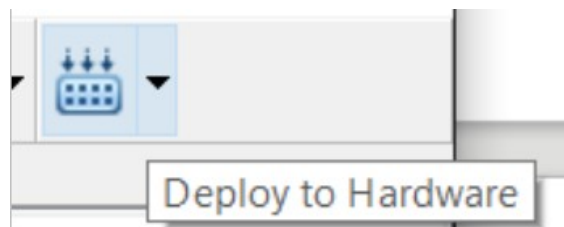
Il faut pour cela installer l'add-on Arduino-Simulink. Depuis l'accueil de Matlab, cliquer sur « Add-Ons » puis « Get Add-ons ». Effectuer une recherche avec le mot clé « Arduino » puis installer le paquetage Simulink pour carte Arduino.

Une fois les « add-ons » pour Arduino installés, ils apparaissent dans la bibliothèque Simulink (figure ci-dessous) :



Il existe deux façons d'exécuter du code généré depuis Matlab/Simulink sur une carte Arduino :

- **Mode « Deploy to Hardware ».**



Dans ce cas, le programme est compilé puis transféré dans l'EPROM de l'Arduino. Il reste alors dans la mémoire jusqu'à ce qu'un nouveau programme le remplace.

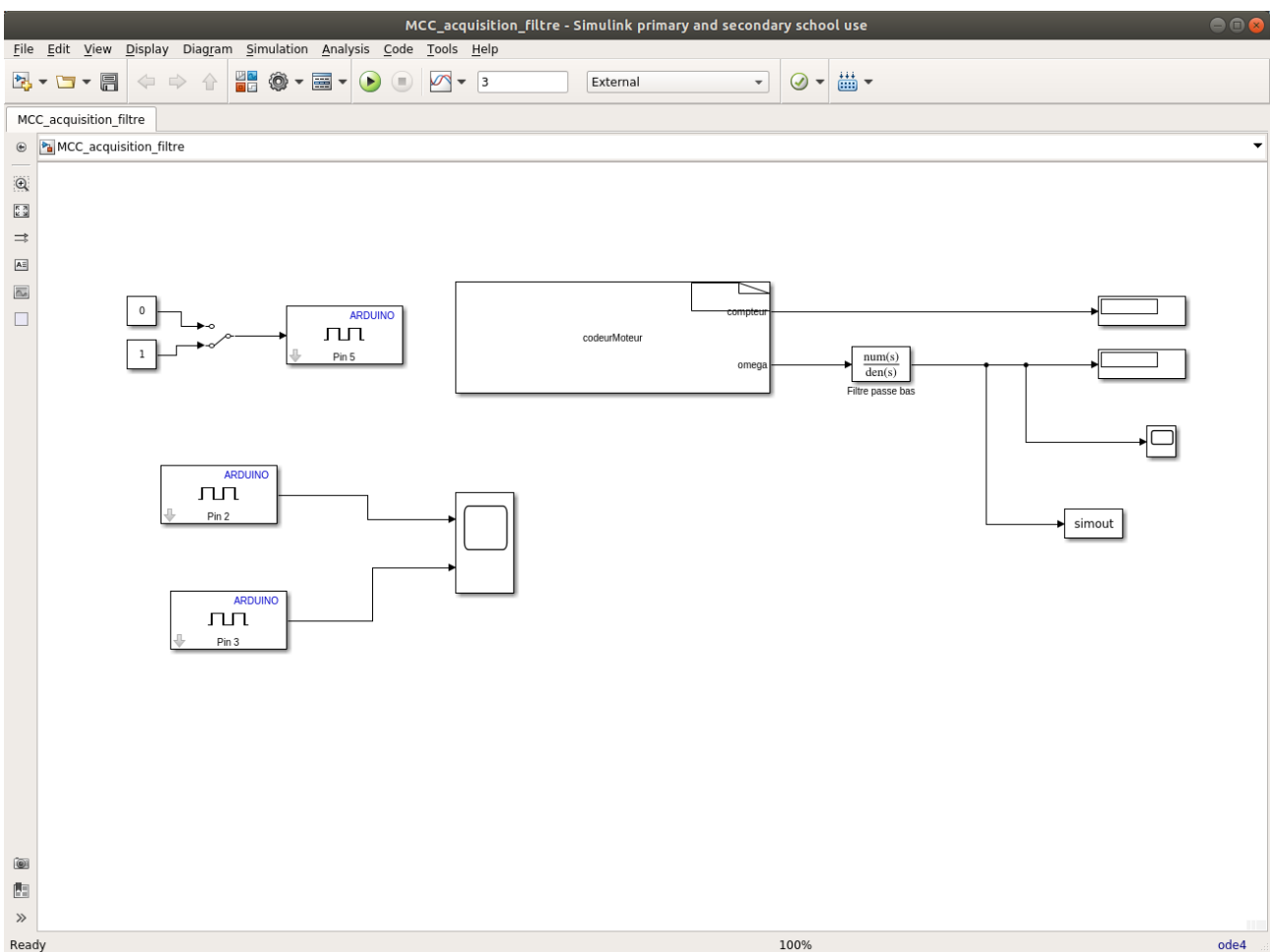
- **Mode « External »**





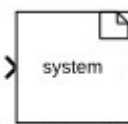
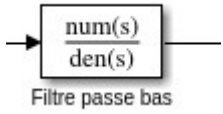
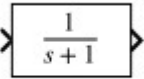
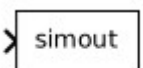


Ce mode permet de lancer l'exécution d'une simulation Simulink directement sur la carte arduino tout en conservant un « monitoring ». On peut alors interagir avec la carte arduino depuis le modèle Simulink.

2.2 Acquisition des données

Pour réaliser l'acquisition de la réponse indicielle, réaliser le modèle ci-dessous. Les blocs nécessaires à la construction du modèle sont indiqués dans le **Tableau 1**.



Nom et rôle du bloc	Bloc Simulink	Bibliothèque
<p>« Constant »</p> <ul style="list-style-type: none"> constante égale à 0 si le moteur est déconnecté constante égale à 1 si le moteur est en rotation 	 <p>Constant</p>	Simulink/Commonly Used Blocks
<p>« Scope »</p> <p>Affiche la réponse indicielle (vitesse de rotation en fonction de temps)</p>	 <p>Scope</p>	Simulink/Commonly Used Blocks
<p>« Digital Output »</p> <p>Met le pin 5 à état « haut » (constante=1) ou « bas »(constante=0)</p>	 <p>Digital Output</p>	Simulink Support Package for Arduino Hardware/Common
<p>« Digital Input »</p> <p>Broches liées aux broches A, respectivement B du codeur</p>	 <p>Digital Input</p>	Simulink Support Package for Arduino Hardware/Common
<p>« S-Function Builder »</p> <p>Bloc qui traite les interruptions et le comptage des impulsions du codeur sur Arduino</p>	 <p>S-Function Builder</p>	Simulink/User-Defined Functions
<p>« Transfer Fcn»</p> <p>Filtre passe défini par sa fonction de transfert (lissage des mesures)</p>  <p>Filtre passe bas</p>	 <p>Transfer Fcn</p>	Simulink/Continuous
<p>« To Workspace »</p> <p>Bloc utilisé pour sauvegarder les mesures dans le « Workspace ». Un autre programme Matlab lira ces données et les écrira dans un fichier Matlab</p>	 <p>To Workspace</p>	Simulink/Sinks

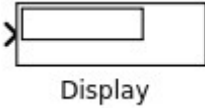
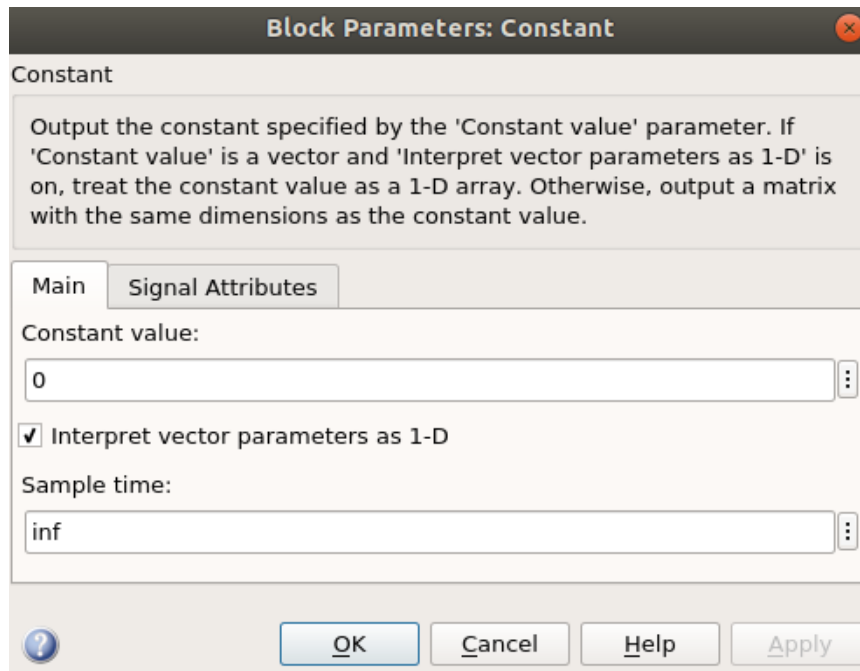
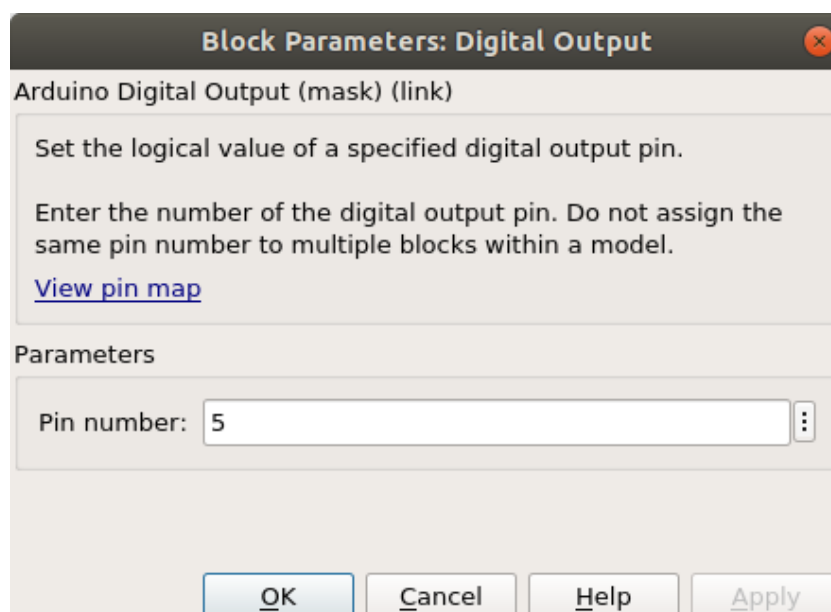
<p>« Display » Bloc pour afficher les valeurs mesurées à chaque instant (nombre d'impulsions et vitesse de rotation)</p>		<p>Simulink/Sinks</p>
--	---	-----------------------

Tableau 1

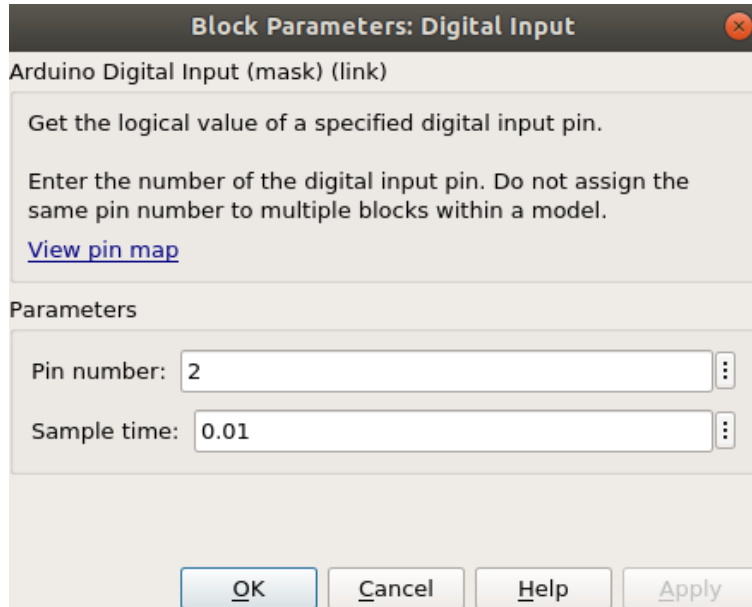
- Paramétrisation du bloc « Constant »



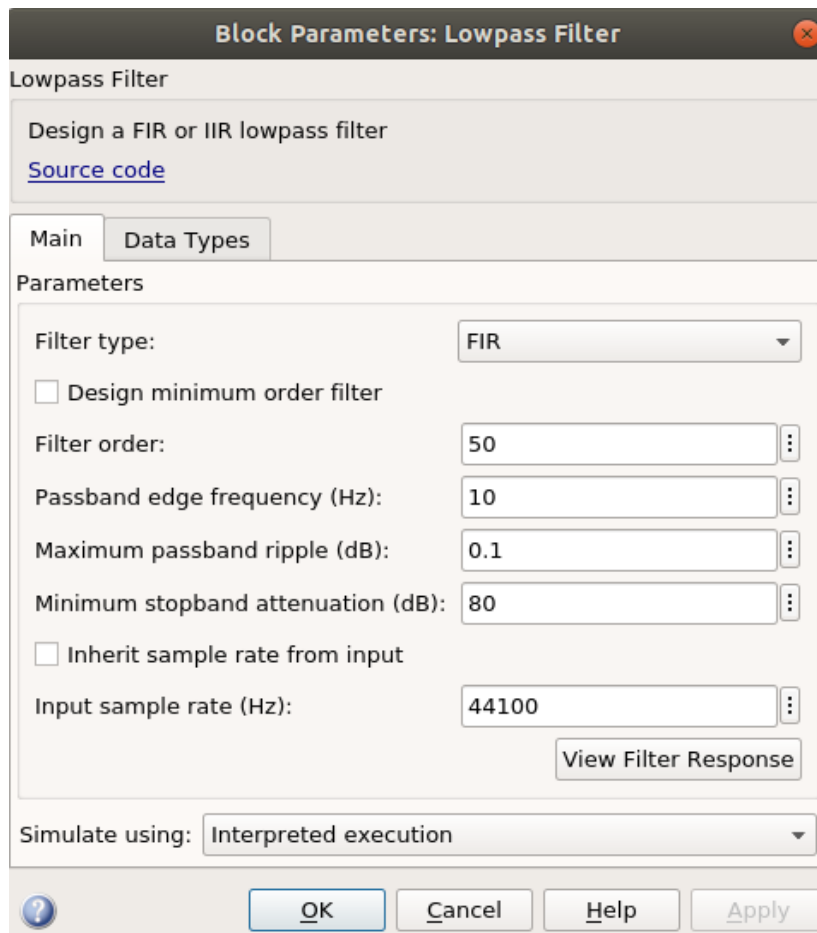
- Paramétrisation du bloc « Digital Output »



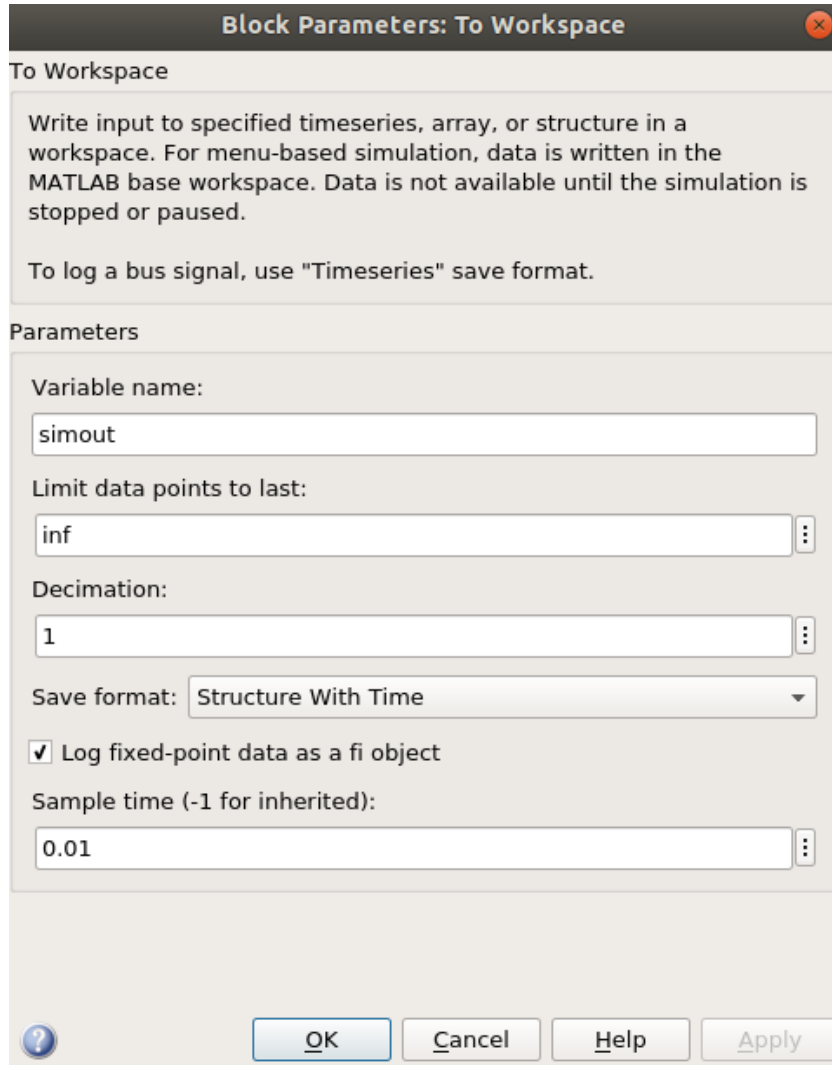
- **Paramétrisation du bloc « Digital Input »**



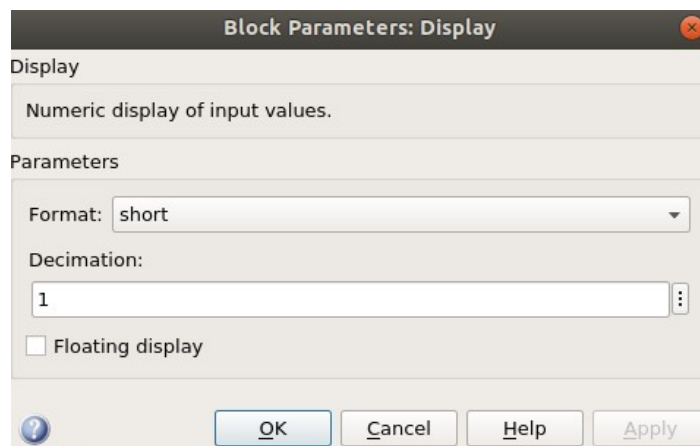
- **Paramétrisation du bloc « Lowpass Filter »**



- **Paramétrisation du bloc « To Workspace »**

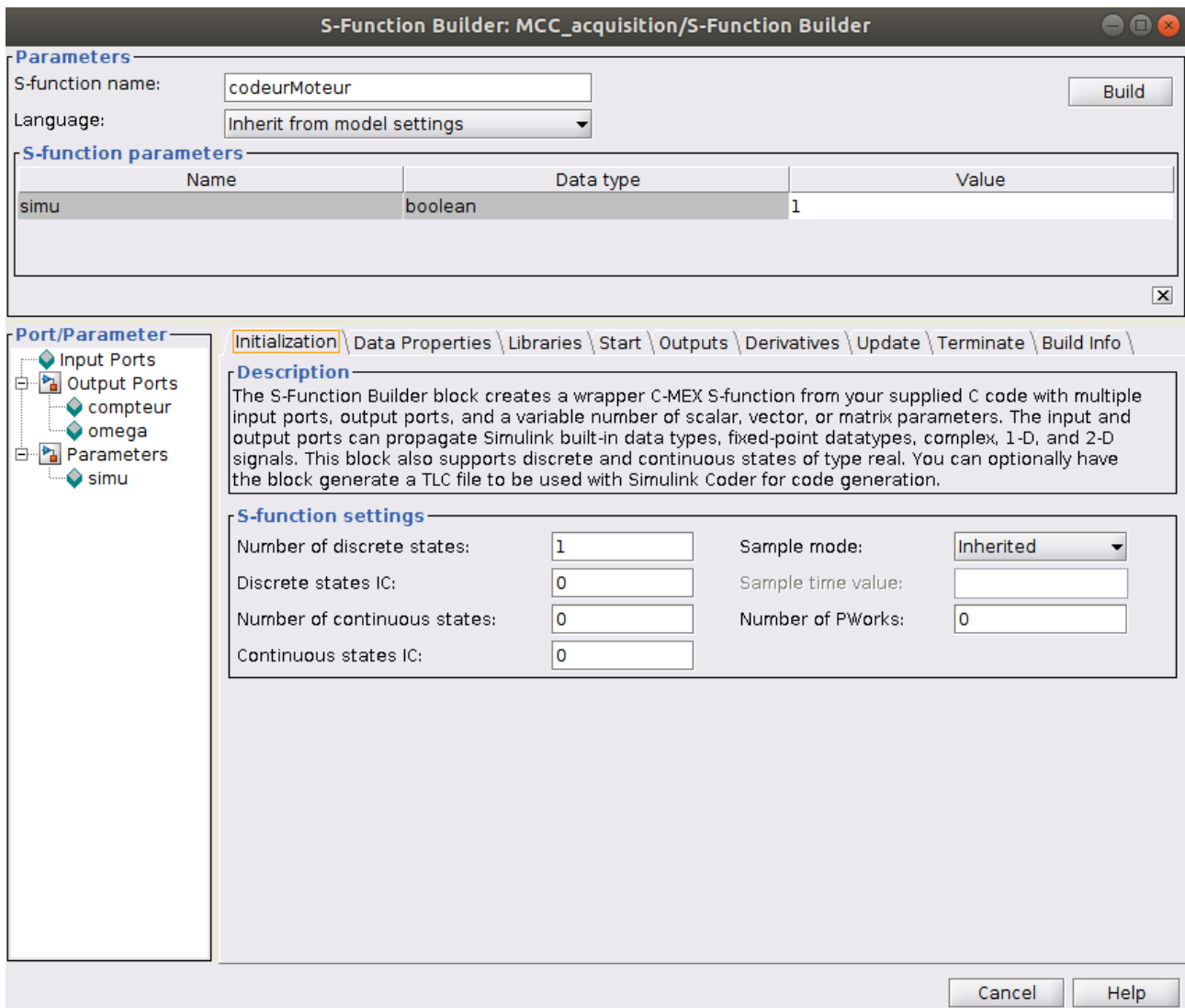


- **Paramétrisation du bloc « Display »**



- **Paramétrisation du bloc « S-Function Builder »**

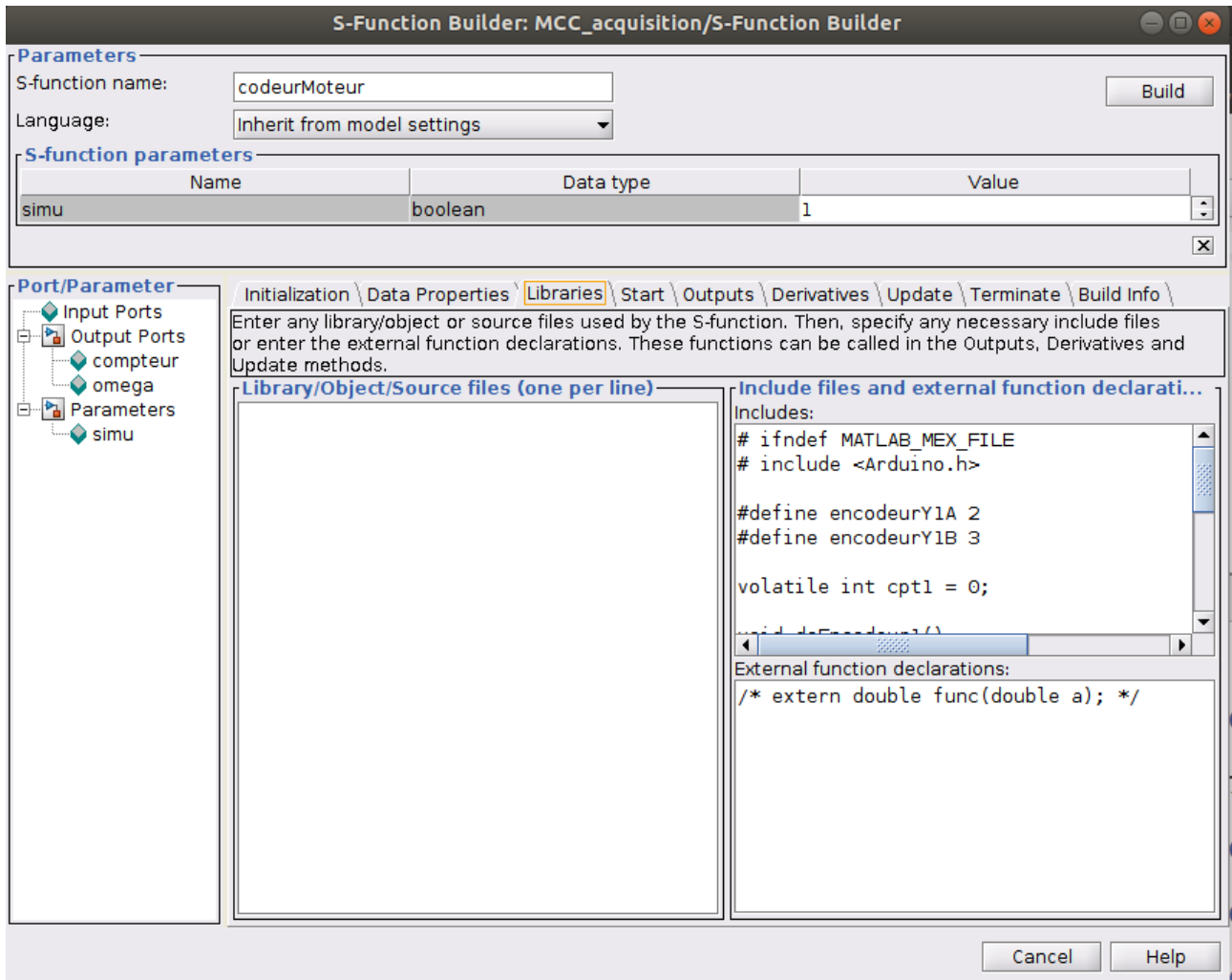
- **Section « Initialisation »**



- **Section « Data Properties »**

Aucune modification à faire.

- Section « Libraries »



Dans fenêtre « Include files and external function declaration... » il faut introduire :

```

# ifndef MATLAB_MEX_FILE
# include <Arduino.h>

#define encodeurY1A 2
#define encodeurY1B 3

volatile int cpt1 = 0;

void doEncodeur1()
{
  if (digitalRead(encodeurY1A) == digitalRead(encodeurY1B))
  {
    cpt1++;
  }
}
else

```

```

{
  cpt1--;
}
}

# endif

```

Il s'agit du morceau de code « Arduino » qui compte les impulsions du codeur.

- Section « Start »

Aucune modification à faire.

- Section « Outputs »

The screenshot shows the 'S-Function Builder: MCC_acquisition/S-Function Builder' window. The 'Parameters' section at the top shows the S-function name as 'codeurMoteur' and the language as 'Inherit from model settings'. Below this is a table for 'S-function parameters':

Name	Data type	Value
simu	boolean	1

The 'Port/Parameter' tree on the left shows 'Output Ports' containing 'compteur', 'omega', and 'simu'. The main area displays the 'Code description' with the following C code:

```

/* This sample sets the output equal to the input
   y0[0] = u0[0];
   For complex signals use: y0[0].re = u0[0].re;
   y0[0].im = u0[0].im;
   y1[0].re = u1[0].re;
   y1[0].im = u1[0].im;
*/
volatile double dt = 0.01;
if (xD[0]==1)
{
  # ifndef MATLAB_MEX_FILE
  compteur[0]=cpt1;
  omega[0] = (60.0*cpt1)/(224.4*dt); // en tours/min
  digitalWrite(simu[0],HIGH);
  //   delay(1);
  //   digitalWrite(simu[0],LOW);
  //   delay(1);
  cpt1=0;
  # endif
}

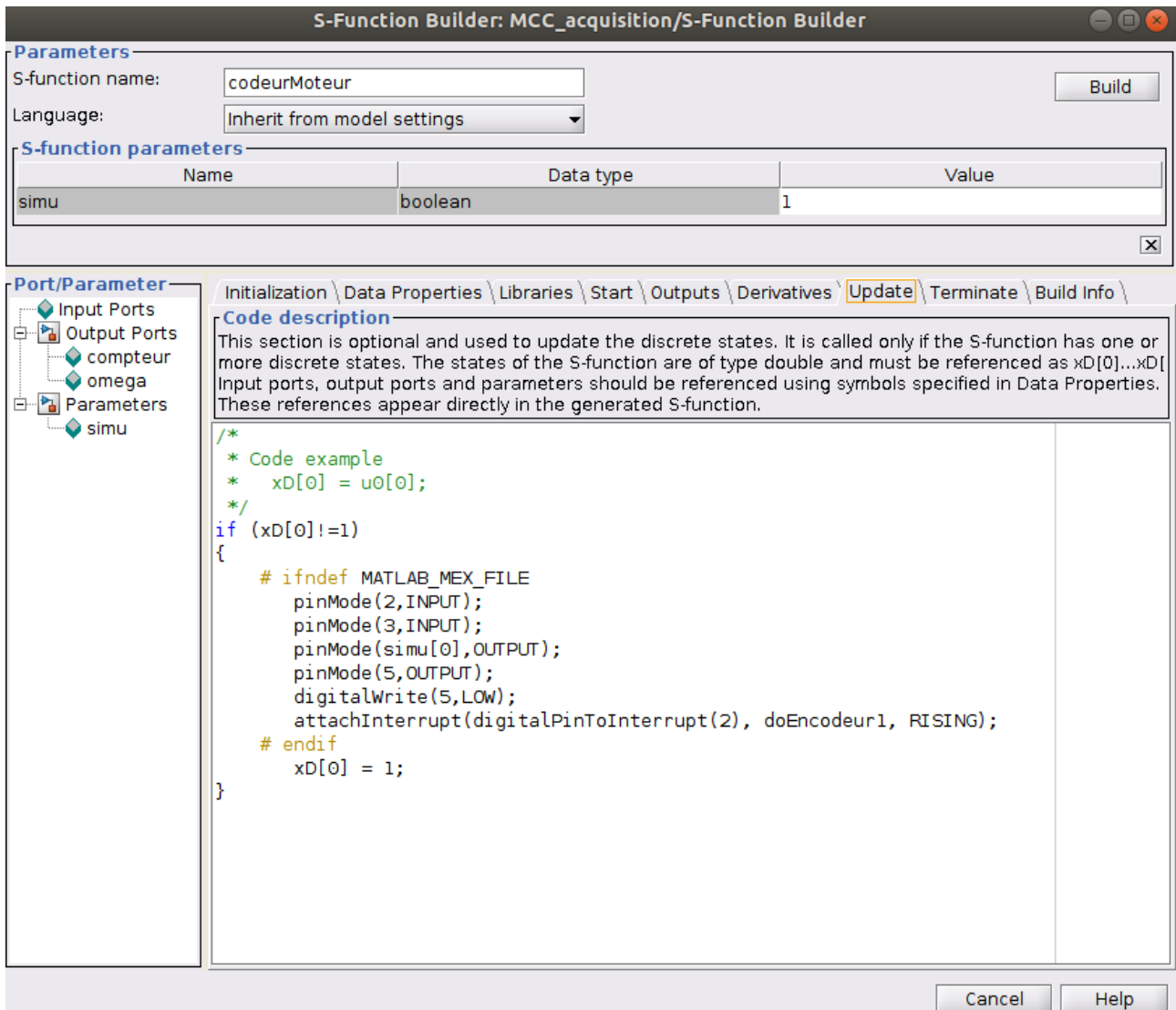
```

At the bottom, there is a checkbox labeled 'Inputs are needed in the output function(direct feedthrough)' which is checked. The window has 'Cancel' and 'Help' buttons at the bottom right.

- [Section « Derivatives »](#)

Aucune modification à faire.

- [Section « Update »](#)



- [Section « Terminate »](#)

Aucune modification à faire.

- [Section « Build info »](#)

Affiche les résultats du « Build ».

Une fois toutes les sections du bloc « S-Function Builder » complétées, il faut créer la bibliothèque correspondante en appuyant sur « Build ».

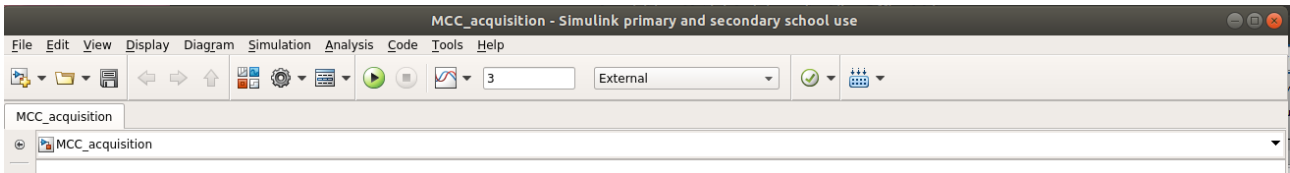
- **Paramétrisation du modèle**

Réaliser le montage Arduino et brancher la carte à l'ordinateur. Dans " Model Parameters" spécifier:

- Le type de simulation : "fixed time step"
- Le pas de temps du modèle : 0.01 s
- Le type de carte: Arduino Méga 2560

Avant de lancer le modèle mettre le bloc " interrupteur" sur le bloc "Constant " de valeur 1. Le moteur doit se mettre en rotation dès que l'acquisition est lancée, afin d'obtenir le plus grand nombre de mesures pendant le régime transitoire.

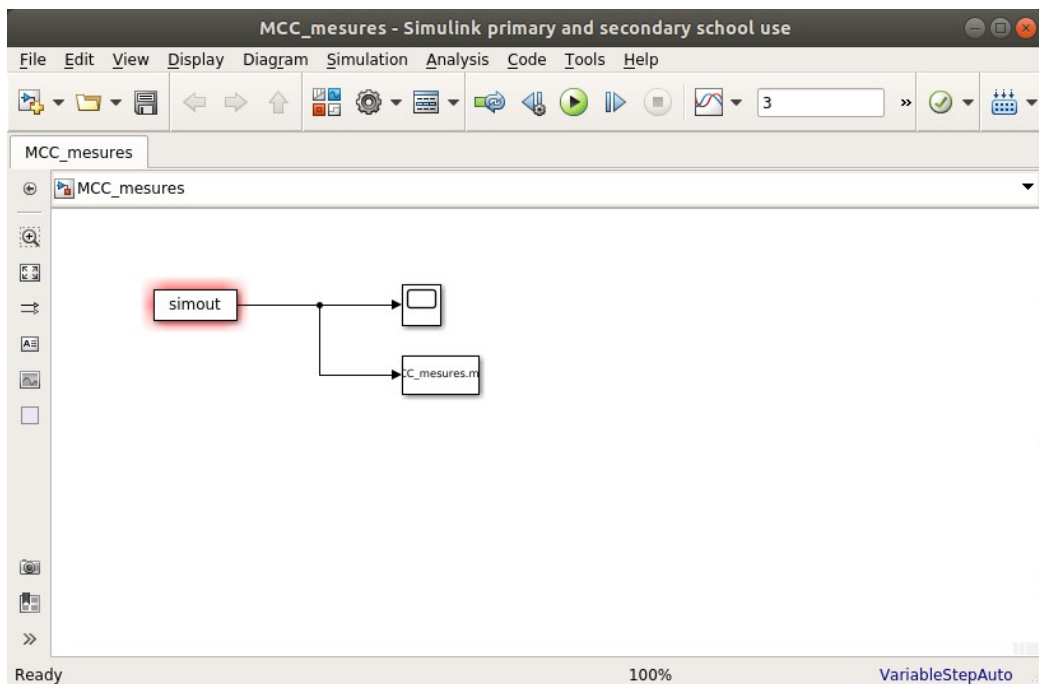
Lancer le modèle en mode "External" pendant 3 s






2.3 Création du fichier de mesures

A la fin des acquisition, dans le "Workspace" on verra apparaître la variable "simout". Les valeurs de cette variable peuvent être écrites dans un fichier compatible Matlab (extension *.mat) pour une utilisation ultérieure. Dans ce fichier les valeurs de la vitesse de rotation du motoréducteur sont enregistrées toutes les 10 ms.

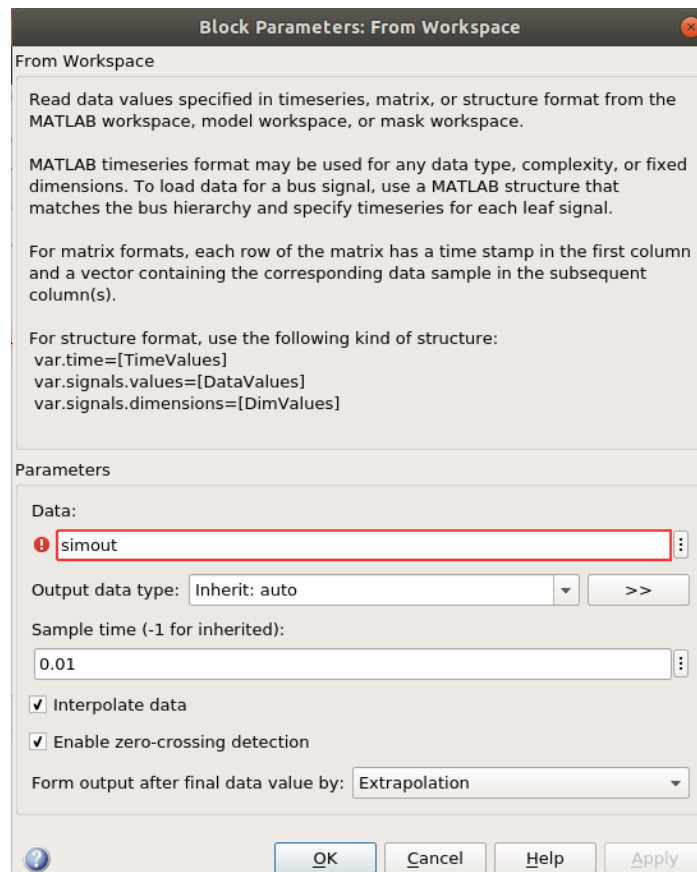
Écrire le modèle ci-dessous pour créer le fichier " mesures.mat".



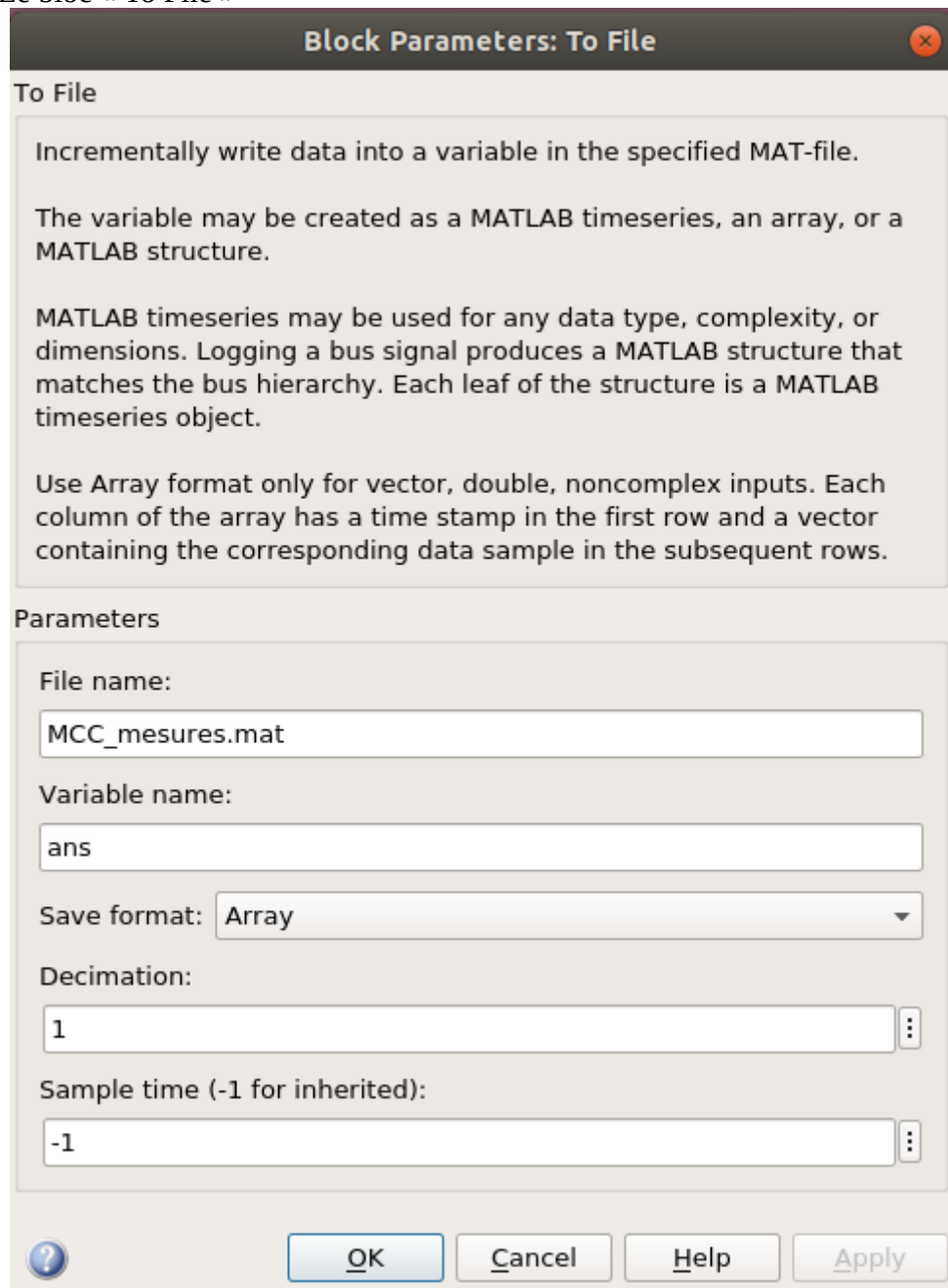
- **Blocs du modèle**

Nom du bloc et rôle	Bloc	Bibliothèque
"From Workspace" Lit la variable "simout" du Workspace qui contient les mesures	 From Workspace	Simulink/Sources
"Scope" Affiche le contenu du "simout" pour vérification	 Scope	Simulink/Commonly Used Blocks
"To File" Écrit les données dans le fichier "mesure.mat"	 To File	Simulink/Sinks

- **Paramétrisation des blocs**
 - **Le bloc « From Workspace »**



- Le bloc « To File »



3. Résultats

Les résultats de l'acquisition sont présentés dans la figure ci-dessous:

