Ecole Nationale d'Ingénieurs de Sousse

Support de TP

Automatique linéaire

Niveau d'étude : 1^{ére} Année informatique appliquée

Réalisé par l'enseignant

ABADI Amine

Table des Matières

TP1 : Prise en main du Matériel Arduino UNO, Logiciel Simulink

1.	Objectif	.1
2.	Présentation et programmation de la carte Arduino	.1
	2.1 Le matériel : Arduino UNO	1
3.	L'interfacage Arduino Matlab/Simulink	.4
	3.1 ArduinoIO	.5
	3.1.1 Pré-chargement du programme dans la carte Arduino	.5
	3.1.2 Installation du package ArduinoIO	5
	3.1.3 Exploitation de la bibliothèque ArduinoIO sous Simulink	.6
4	Acquisition des données	.7
	4 1 Présentation du ADC	.7
	4.2 Acquisition des données : Capteur de température LM35	.7
	4.2.1 Présentation du capteur	.7
	4.2.2 Branchement avec la carte Arduino UNO	.8
	4.2.3 Exploitation du package ArduinoIO Library	. 8
5.	Envoi des données	. 8
0.	5.1 Présentation des sorties analogiques (mode PWM)	.8
	5.2 Commande d'une résistance chauffante	10
	5.2.1 Présentation du schéma électronique	10
	5.2.2 Exploitation du package ArduinoIO Libraray	10
тр)	· Modélisation et Analyse du Processus Thermique	
II 	Objectife	11
1.	Drécentation de la maguatte	11
2. 2	Modélisation du procédé thermique	11
5.	2 1 Présentation de l'étane d'identification avec Matlab	12
	3.1 Fresentation de la rénonse indicielle du système	12
	3.2 Acquisition de la fonction de transfert H(n)	13
	5.5 Determination de la fonction de transfert fi(p)	15
143	: Commande d'un système thermique	
1.	Objectifs	17
2.	Présentation de la maquette	17
3.	Utilisation du PID TUNING pour la régulation du système	17
	3.1 Synthèse du régulateur numérique	18
	3.2 Implémentation de la commande sous Simulink	19
4.	Travail à faire	20
TP4	: Asservissement du processus thermique	
1.	Objectifs	21
2.	Présentation de la maquette	21
3.	Modélisation du procédé thermique	21
4.	Implémentation de la commande sur la carte arduino	22
	4.1. Le correcteur à retard de phase	22
	4.2 Implémentation de régulateur	23
5.	Travail à faire	24

TP1 : Prise en main du Matériel Arduino UNO, Logiciel Simulink

1 Objectifs

- Familiarisation avec la carte Arduino UNO.
- Réalisation de l'interfaçage avec Matlab/Simunlink.
- Acquisition des données via capteur de température.
- Commande PWM d'une résistance chauffante.

2 Présentation et programmation de la carte Arduino

Arduino est un projet créé par une équipe de développeurs composée de six individus : Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, David Mellis et Nicholas Zambetti. Cette équipe a créé le "système Arduino". C'est un outil qui va permettre aux débutants, amateurs ou professionnels de créer des systèmes électroniques plus ou moins complexes

2.1 Le matériel : Arduino UNO

C'est un circuit imprimé comportant tous les composants électroniques nécessaires pour faire fonctionner un microcontrôleur (Atmega 328) associé à une interface USB lui permettant de communiquer avec un ordinateur.



2 GND, Source 5V – 3,3V Analog Input Figure 1 – Description de la Carte Arduino "Uno"

- Microcontroller : ATmega328
- Operating Voltage : 5v
- Input Voltage (recommended) : 7-12 v
- Input Voltage (limits) : 6-20 v
- DC Current per I/O Pin : 40mA
- DC Current for 3.3V Pin :50mA
- Flash Memory :32 KB
- Clock Speed : 16MHz
- Pins assignements :
- Analog read(A0-A5)
- Analog write[PWM] (3,5,6,9,10,11)
- Digital read (2-19)
- Digital write (2-19)

2.2 Le logiciel Arduino

Arduino IDE (Integrated Development Environment). Le logiciel est gratuit et open source dont la simplicité d'utilisation est remarquable. Ce logiciel va nous permettre de programmer la carte Arduino pour :

– Réaliser l'interfaçage avec Matlab/simulink

- Implémenter la commande directement sur la carte.

Lancer le logiciel Arduino



Figure 2 – L'interface du logiciel Arduino

Relier la carte Arduino UNO à votre ordinateur à l'aide du câble US



Sélectionner la carte Arduino UNO sur le logiciel Arduino



Sélectionner le port de communication utilisé par votre machine pour dialoguer avec la carte Arduino UNO

Sketch_oct29a Arduino 1. Fichier Édition Croquis Du	5.4 His Aide			x	
sketch_oct28a	Formatage automatique Archiver le croquis Réparer encodage & recharger	Ctrl+T			
// put your setup	Moniteur sèrie Board	Ctrl+Maj+M			
F	Port	•		CO	V 16
<pre>void loop() { // put your main</pre>	Programmateur Graver la séquence d'initialisation	,		C01	47 410
>			~	cor	V112
				COL	M13
				COI	414
				COL	//20
				COL	M21
				COL	//22

Editer le programme blink. (Fichier/Exemples/01 Basics/Blink). Ce programme fait clignoter la DEL L

T IC	hier Edition Croquis Outlis Aide					
	Nouveau Ouvrir Camet de croquis Exemples Fermer	Ctrl+N Ctrl+0 Ctrl+W	•	01.Basics 02.Digital 03.Analog 04.Communication		AnalogReadSerial BareMinimum Blink DigitalReadSerial
	Enregistrer Enregistrer sous Téléverser Téléverser avec un programmateur	Ctrl+S Ctrl+Maj+S Ctrl+U Ctrl+Maj+U		05.Control 06.Sensors 07.Display 08.Strings 09.USB	•	Fade ReadAnalogVoltage
	Imprimer	Ctrl+Maj+P Ctrl+P		ArduinelSP	-	
	Quitter	Ctrl+Q		EEPROM Esplora Ethernet		

Compiler le programme blink



Télécharger le programme blink dans la carte Arduino UNO



Analyse de programme

/* Blink Turns on an LED on for o This example code is in th */	ne second, then off for one second, repeatedly. e public domain.	/* presentation générale du programme
<pre>// Pin 13 has an LED conne // give it a name: int led = 13;</pre>	cted on most Arduino boards.	// commentaires Déclaration de la variable led de type int. On donne la valeur 13 à la variable led. Ainsi le terme led est équivalent à la valeur 13 Rappelons que la led L est connectée à la broche 13 de la carte
<pre>// the setup routine runs ond void setup() { // initialize the digital pin pinMode(led, OUTPUT); }</pre>	e when you press reset: as an output.	// commentaires Void setup() Configuration des entrées/sorties pinMode(led, OUTPUT) = pinMode(13, OUTPUT) car la variableled=13. La broche digitale 13 est paramétrée en sortie http://arduino.cc/en/Reference/pinMode
// the loop routine runs over	r and over again forever:	// commentaires
<pre>void loop() { digitalWrite(led, HIGH); delay(1000); digitalWrite(led, LOW); delay(1000); }</pre>	// tum the LED on (HIGH is the voltage level) // wait for a second // tum the LED off by making the voltage LOW // wait for a second	Void loop(), instructions exécutées en boucle digitalWrite(led, HIGH)= digitalWrite(13, HIGH) car la variable led=13 après cette instruction la broche 13 est positionnée au niveau haut de tension la DEL s'allume http://arduino.cc/em/Reference/DiatalWrite delay(1000): faire une temporisation de 1000ms la diode restera allumée 1s http://arduino.cc/em/Reference/Delay. digitalWrite(led, LOW); la broche 13 est positionnée au niveau bas de tension la DEL s'éteint delav(1000): faire une temporisation de 1s la diode s'éteint 1 s

Nous allons maintenant réaliser le schéma suivant :



```
Serial.begin(9600);

}

void loop() {

digitalWrite(10, HIGH); // allume la LED

delay(500); // attend 500ms

digitalWrite(10, LOW); // éteint la LED

delay(500); // attend 500ms

}
```

On ajoute une autre led connectée à la branche 12

-Modifier le code précédent pour faire clignoter les deux Leeds en même temps.

-Modifier le code pour faire clignoter les deux Leeds en alternance.

-Ecrire un programme qui permet de clignoter une LED 10 fois.

3 L'interfaçage Arduino ← Matlab/Simulink

Il existe trois possibilités d'interfacer la carte Arduino avec Matlab/Simulink, à savoir :

- 1. Programmation de la carte Arduino Uno comme une carte d'interface.
- 2. Utilisation du package ArduinoIO.
- 3. Utilisation du package Arduino Target.

Dans la suite on va utiliser la deuxième méthode d'interfaçage.

3.1 ArduinoIO

Cette solution consiste à utiliser la carte arduino comme une interface d'entrées (Analog Input) /sorties (Analog/Digital Output). Ce package permet de communiquer Matlab ou Simulink avec la carte Arduino via un câble USB. Elle consiste à pré-charger un programme dans la carte Arduino afin que celle-ci fonctionne en serveur.

Ce programme consiste à "écouter" les requêtes envoyées via la liaison série (USB) et de répondre à ces requêtes en renvoyant l'état d'une entrée ou en modifiant l'état d'une sortie. Ces mêmes entrées/sortie sont vues dans Matlab comme des entrées logiques ou analogiques(utilisation du CAN) ou des sorties analogiques (mode PWM).

3.1.1 Pré-chargement du programme dans la carte Arduino

- 1. Télécharger le package ArduinoIO
- 2. Décompresser à la racine de votre disque dur, exemple E : \arduino
- 3. Ouvrir le dossier décompressé.
- 4. Aller vers : "ArduinoIO\pde\adiosrv" *
- 5. Charger le fichier *adiosrv.pde* vers le logiciel Arduino.
- 6. Televerser !

* adiosrv est l'abréviation de : Analog and Digital Input and Output Server for MATLAB. La carte Arduino UNO est maintenant configurée pour être utilisé comme une carte d'interface Entrées/Sorties.

3.1.2 Installation du package ArduinoIO

- 1. Lancer Matlab2013 et placer vous dans le répertoire E : \arduino
- 2. Exécuter la commande : install_arduino
- 3. Fermer et relancer Matlab puis Simulink

4. Dans les bibliothèques se trouvent maintenant les blocs dans Arduino IO library.

	Simulink Library Browser –								
File Edit View Help									
💀 🗀 🔹 power 🗸 🚧 🔍									
Libraries	Library: Arduino IO Library	Found: 'power' Frequen	tly Used						
P Simulink A Pia Acrounce Nockset A Pia Acrounce Nockset A Pia Acrounce Nockset A Pia Communications System Toolbox D Pia Computer Vision System Toolbox D Pia Control System Toolbox D Pia Control System Toolbox D Pia Control System Toolbox D	Arduino Analog Read Analog Read Arduino Digital Write	Arduino Analog Write Analog Write Analog Write Analog Write	Arduino Digital Read						
Barbedded Coder Pa Fuzzy Logic Toolbox Pa HDL Verifier	Encoder Read (#1) Encoder Read	Arduino1 Encoder Reset (#0) Encoder Reset	Real-Time Pacer Speedup - 1 Pacer						
 Image Acquisition Toolbox Instrument Control Toolbox Model Predictive Control Toolbox 	Arguinot Bervo Read (W) Servo Read	Addunot Bervo Write (#9) Servo Write	Stepper Motor						

Figure 3 – ArduinoIO Library



Figure 4 – Les Blocs d'ArduinoIO nécessaires pour la commande

3.1.3 Exploitation de la bibliothèque Arduino IO sous Simulink

Les blocs nécessaires pour notre objectif d'asservissement sont les suivants :

– Real-Time Pacer : Ce bloc permet de ralentir le temps de simulation de sorte qu'il synchronise avec le temps réel écoulé. Le coefficient de ralentissement est contrôlable par L'intermédiaire du paramètre *Speedup*.

Arduino IO Setup : Pour configurer sur quel port la carte Arduino UNO est connectée.
Pour cela il suffit de voir dans Gestionnaire des périphériques. Voir Figure 4.



Figure 5 – Emplacement COM de la carte Arduino UNO

- **Arduino Analog Read** : Pour configurer à partir de quel pin [0,1,2,3,4,5] on va acquérir les données du capteur.

- Arduino Analog Write : Pour configurer à partir de quel pin [3,5,6,9,10,11] on va envoyer la commande en PWM vers l'actionneur.

4 Acquisition des données

4.1 Présentation du ADC

La carte Arduino Uno dispose de 6 entrées analogiques notées A0, A1,..A5 mais d'un seul convertisseur analogique/numérique, la durée d'une conversion est de l'ordre de 100µs.Il a une résolution de 10 bits. La donnée numérique qu'il fournit après conversion est donc comprise entre 0 et 1024



Figure 6 – Type du CAN de la carte Arduino UNO

4.2 Acquisition des données : Capteur de température LM35

4.2.1 Présentation du capteur

Le LM35 fait partie des capteurs de température électroniques de précision en structure intégrée.



Figure 7 – Capteur de température LM35

D'après la fiche technique :

$$10mV \rightarrow C^o$$

Autrement un volt correspond à 100 degrés Celsius. La lecture analogique d'un signal de 0 à 5V étant codée de 0 à 1023, on a la formule :

$$Temp = Volt * (5/1023) * 100;$$
 Volt.entre.(0et1023)

4.2.2 Branchement avec la carte Arduino UNO

Pour exploiter le capteur LM35, il suffit :

- D'alimenter les pattes VCC et GND
- Débrancher la patte centrale à une entrée analogique d'Arduino (A0,...,A5).

4.2.3 Exploitation du package Arduino IO Library

- 1. Pré-chargement de adiosrv.pde sur la carte Arduino UNO
- 2. Développement du modèle Simulink



Figure 8 - Branchement du Capteur LM35 avec Arduino UNO



Figure 9 – Acquisition de la température sous Arduino IO Library

5 Envoi des données

5.1 Présentation des sorties analogiques (mode PWM)

La carte Arduino Uno dispose de 6 sorties (3,5,6,9,10 et 11) qui peuvent être utilisées en mode PWM, c'est-à-dire en modulation de largeur d'impulsion. Ce sont des signaux logiques binaires de fréquence constante (500Hz) mais de rapport cyclique variable.



Figure 10 – Description du signal PWM

Lorsqu'un moteur ou une lampe est alimenté par ce type de tension, tout se passe comme s'il était alimenté par une tension continue ajustable entre 0V (rapport cyclique=0) et 5V (rapport cyclique=255). Ces sorties doivent être initialisées comme des sorties digitales.

$$V_{out} = V_s imes rac{ au_o}{ au_c}; \qquad avec: \ \ au_c = 2ms$$



Figure 11 – Exemples de variation du rapport cyclique

Les composants utilisés sont les suivants :

– Le transistor TIP121 : C'est un transistor Darlington NPN qui d'après la fiche technique permet d'amplifier le courant jusqu'à 5A avec son gain d'amplification "au minimum" $\beta = 1000$ et supportant.

La diode 1N4004 : Dans une charge inductive (bobines), le courant ne peut pas se stopper
 Instantanément. Cette diode joue le rôle d'une diode de roue libre qui permet au courant
 de s'arrêter progressivement.

5.2 Commande d'une résistance chauffante

5.2.1 Présentation du schéma électronique

Le circuit électronique comporte :

- Une lampe halogène 12V-35W
- Une résistance $1k\Omega$
- Un transistor TIP121



Figure 12 – Branchement de la carte Arduino UNO avec une lampe

L'utilisation de la commande PWM à partir de la carte Arduino permet de faire varier la tension appliquée aux bornes de la lampe autrement ceci permet de contrôler l'intensité lumineuse de la lampe.

5.2.2 Exploitation du package ArduinoIO Libraray

- 1. Pré-chargement de adiosrv.pde sur la carte Arduino UNO
- 2. Développement du modèle Simulink



Figure 13 – Envoi de la commande PWM sous ArduinIO Library

TP2 : Modélisation et Analyse du Processus Thermique

Dans la conception d'un système d'exploitation industriel, on accorde une grande importance à l'identification des processus. En effet, pour procéder à la commande d'un système il est nécessaire de disposer d'un modèle mathématique caractérisant le système réel avec une bonne approximation mais suffisamment simple pour faciliter l'étude.

1 Objectifs

- Déterminer le modèle du processus thermique
- Déterminer les paramètres caractéristiques du processus.
- Analyser la stabilité du processus.
- Réaliser un bouclage simple avec une commande proportionnelle.

Remarque :

-Vous allez réaliser le TP en utilisant la polycopie intitulée" Notions Automatiques" donnée au début du TP

-Un compte rendu avec les résultats intermédiaires est à rendre à la fin de la séance par chaque groupe d'étudiants.

2 Présentation de la maquette

La maquette est constituée d'un capteur de température LM35 et une Lampe halogène 12V 35W qui joue le rôle d'un élément chauffant. Le capteur et la lampe sont installés dans une boite en bois avec un couvercle en plexiglas. Cette boite représente le système thermique à commander. La figure suivante schématise la connexion de la carte Arduino UNO avec l'entrée et la sortie du système thermique.





La figure ci-dessous montre une vue réelle de la maquette utilisée.



Figure 2 – Une vue de la maquette

3 Modélisation du procédé thermique

Le but de cette partie est de déterminer la fonction de transfert continu de notre procédé thermique en boucle ouverte notée H(p). L'entrée du système est la tension u(p) en *volts* et la sortie est la température T(p) de *degré celsius*.

3.1 Présentation de l'étape d'identification avec Matlab

Cette étape est constituée de deux parties. La première est assurée par l'environnement Simulink et le package Arduino IO pour l'envoi et l'acquisition des données. La deuxième partie est assurée par l'outil *System identification* sous Matlab.



Figure 3 – L'utilisation de l'outil System Identification

3.2 Acquisition de la réponse indicielle du système

Plusieurs méthodes sont utilisées pour la modélisation d'un système comme la détermination des équations physiques du système, l'étude de la réponse d'un système à une entrée...etc. Dans notre cas on va identifier notre système en étudiant la réponse de notre système à échelon de tension.

Le modèle Simulink permettant de réaliser l'acquisition de la réponse du système à un échelon de tension est le suivant :



Figure 4 – Modèle Simulink pour la détermination de la réponse indicielle

3.3 Détermination de la fonction de transfert H(p)

Après avoir déterminé la réponse du système, on passe à la détermination de la fonction de transfert

1. Ouvrir l'outil System identification Tool

e Options Window	Help		
Import data	Operations	Import models	
	< Preprocess		
Data Views	Estimate> V To To Workspace LTI Viewer Mod	Model Views del output Transient resp	Nonlinear ARX
Data spectra	Trash Validatio	Del resids Frequency resp Zeros and poles	Hamm-Wiener

Figure 5 – L'interface de l'outil System identification

2. Cliquer sur *import* data et choisir *Time domain data*.

Import data	✓ Operations	Import models V	
Time domain data Freq. domain data Data object Example			
	Estimate>		
Data Views	Estimate> V	Model Views	
Data Views	Estimate> V To Workspace LTI Viewer Model	Model Views	Nonlinear ARX
Data Views	Estimate> V To To To Workspace LTI Viewer Model Model	Model Views I output Transient resp I resids Frequency resp	Nonlinear ARX

Figure 6 – Choix des types de données "Time Domain Data"

3. Entrer le nom de la variable Input et la variable Output ainsi que le temps de starting time et sample time qu'on va utiliser lors de l'identification avec Simulink. Enfin cliquer sur Import.

Import Da	
Data Forr	mat for Signals
Time-Domain S	Signals 🗸 🗸
Works	oace Variable
Input:	
Output:	
Data	aformation
Data I Data name:	nformation
Data I Data name: Starting time	nformation mydata
Data I Data name: Starting time Sampling interval	nformation mydata
Data I Data name: Starting time Sampling interval	nformation mydata 1 1 More
Data I Data name: Starting time Sampling interval	nformation mydata 1 1 More Reset

Figure 7 – Saisi des données relatives aux Input et Output du système

4. Cliquer sur Estimate et choisir "Transfer Function Models"



Figure 8 – Choix de la description du système à estimer"Tranfer Function"

5. Entrer le nombre de pôles et de zéros et cliquer sur Continuous-Time ensuite cliquer sur Estimate



Figure 9 - Choix du nombre des pôles et des zéros de la fonction de transfert à estimer

6. Revenir à l'interface System Identification Tool et cliquer deux fois sur tf1



Figure 10 – Visualisation du résultat de l'estimation

7. Une fenêtre apparait dans laquelle vous trouvez H(p).

Model name:	tf1	
Color:	[0,0,1]	
10m input 0.004315	of " to output "y1":	^
1 - 0.9994	 z^-1	
lame: tfl		
Sample time:	seconds	
Sample time: Nicercto-time K	identified transfer fun	~~ ×
Sanghe time: Disarata-time	Diary and Notes	~÷` *
Sangha time: Disaratatim	Diary and Notes	~;; ×
Sample time: Discontation	Diary and Notes	^
Sample time: Nicerote-time <	Diary and Notes	~;; ~ ^ ~

Figure 11 - Récupération de la fonction de transfert estimée

TP3 : Commande d'un système thermique

Pour améliorer la qualité de l'asservissement, on se propose de mettre en place une structure de commande plus évoluée que la correction proportionnelle pure.

L'objectif de ce TP est de réguler un système de premier ordre (un procédé thermique). Après l'identification du système en TP2, vous allez dans ce TP étudier les caractéristiques du système en boucle fermée en appliquant plusieurs correcteurs (P,I,PI,PD,PID).

MATLAB sera utilisé durant ce TP comme outil d'analyse des relevés expérimentaux mais permettra également d'accélérer la phase de synthèse du correcteur avec la toolbox PID TUNING.

1. Objectifs

- Familiariser avec l'outil PID TUNING.

-Etudier l'effet des régulateurs (P,I,PI,PD,PID) sur la réponse du système en utilisant le PID tunning

2. Présentation de la maquette

La maquette est constituée d'un capteur de température LM35 et une Lampe halogène 12V 35W qui joue le rôle d'un élément chauffant. Le capteur et la lampe sont installés dans une boite en bois avec un couvercle en plexiglas. Cette boite représente le système thermique à commander. La figure suivante schématise la connexion de la carte Arduino UNO avec l'entrée et la sortie du système thermique.



Figure 1 – Branchement du procédé avec la carte Arduino

3. Utilisation du PID TUNING pour la régulation du système

L'étape de la commande du procédé thermique avec le PID TUNING est constituée de deux parties.

1. La première partie consiste à utiliser l'outil Matlab *PID Tuning* pour déterminer les différents paramètres de notre régulateur PID à savoir Kp,Ki et KD en fonction de notre objectif de commande. 2. La deuxième partie consiste à implémenter les paramètres du régulateur sous Simulink. PID(z).

				MATLAB	R2013a				
HOME PLOTS	APPS	VA	RIABLE		<u>لما</u>	stear and little are	b 122 922 💭	② Search Docum	nentation 🔎
Let More Install Package Apps App App	Curve Fitting	Optimization	MuPAD Notebook	PED Tuning	System Identification	Signal Analysis	Image Acquisition	Instrument S Control	amBiology
	Program Files		R2013a > bin		APPS				
urrent Folder 💿	T Variables	- EBO					() ×	Workspace	
Name - Worker.bat	: s8 × e8	× BO × I x1 double>	BOK					Name ~	Value <6498x1 doub
mexulis.pm messetup.pm mexsetubat mex.pl mex.bat MemShieldStarter.bat mc.bat	1 2 3 4 5 6	2 6 6 6 6 6 6 6	3	A	5	6	7	EBO ans e8 s8 tout u8	<6498x1 doub 0 <1708x1 doub <1708x1 doub <1000x1 doub <1708x1 doub
 matlab.exe matlab.bat lcdata_utf8.xml lcdata.xsd 	7 8 9 <	6						Command Hi	story
instype.ini deploytool.bat win64 util registry m3iregistry	Command V >> plot >> plot >> EBO= >> EBO= /ft >>	Window (88) (88) (80) 6*ones (1, 6 6*ones (649)	198); 1,1);				C	<pre></pre>	/2014 23:071 8) 8) 9) 0) 0) 0) 0) 0) 0) 0) 0) 0) 0

Figure 2 – Emplacement de l'outil PID tuning

3.1 Synthèse du régulateur numérique

1. Ouvrir l'outil PID Tuner



Figure 3 – Interface de l'outil "PID tuning"

2. Cliquer sur Import new plant, une nouvelle fenêtre apparait.



Figure 4 – Importation du modèle estimé

3. Une nouvelle fenêtre apparait dans laquelle vous allez sélectionner **tf1** ensuite cliquer sur *import* puis *close*.

	Import Linea	r System	×
Import a linear system Import from:	as: Plant model	Browse	~
A itale Data tf1	Type tf	Order 1	
Specify the number of	unstable poles (except	integrators) for the selecte	ed plant:

Figure 5 – Interface de l'outil "Import Linear System"

4. Revenir à la fenêtre PID Tuner, vous pouvez choisir le type de régulateur à implémenter et les objectifs de la commande en boucle fermée et voir la réponse de la sortie du système.



Figure 6 – Choix du régulateur à implémenter

5. Cliquer sur la flèche de show paramètre pour voir les paramètres utilisés de votre régulateur ainsi que les performances du système en boucle fermée.

2	PID	Tuner		- • 💌
월 월 · 독 · 국 · 안 드리 · 양 · Design mode Time domain	v Form: Parallel v Type: P v			
Plot: Step y Response Reference tracking	v Show baseline			Hide parameters
97			Countrier parameters	
0.6			Ko	Tuned 0.32012
03			Kd TY	
2 0.4			Performance and robusilia	
§ 03			Rise time Setting time	Tuned 555 seconds 955 seconds
0.1			Overshoot Peak	0.5607
10 ten	1000 1500	Tured response	Phase margin Closed-loop stability	116 deg @ 0.00249 rad/s Stable
	Time (seconds)			
Response time:				
(4) Slover				Faster 803 seconds •
0				Close Help

Figure 7 – Récupération des paramètres du régulateur

3.2 Implémentation de la commande sous Simulink

La boucle d'asservissement à implémenter sur Simulink se traduit par le schéma suivant :



Figure 8– Synoptique de la boucle d'asservissement à implémenter

L'asservissement de notre procédé est assuré par le schéma Simulink ci-dessous qui regroupe la consigne, le comparateur, le correcteur PID(z), le traitement de la température issue du capteur et l'envoi de la commande PWM.



Figure 9 – Modèle Simulink d'asservissement de température

L'appui deux fois sur le bloc PID(z) permet d'introduire les paramètres Kp Ki Kd et de configurer le régulateur selon notre objectif de commande.

1		Fund	tion Block Parame	ters: Disc	rete PID Co	ntroller		×
PID Controll This block in windup, exte Simulink Cor	er nplements ernal reset, ntrol Desigr	continuous- and and signal tracl n).	discrete-time PID cor king. You can tune the	ntrol algor PID gains	ithms and inc automatical	ludes advance ly using the 'Ti	ed features such as une' button (requ	s anti- Jires
Controller: P	ID)		• Forn:	Parallel)		•
Time domonio O Continuous-time O Discrete-time			Disc Integ Filte Sam	grator method: r method: pple time (-1 t	tings d: for inherite	Forward Euler Forward Euler 0.5	•	
Main PII Controller J Proportiona Integral (I) Derivative (Filter coeffi	D Advanced parameters of (P): : D): cient (N):	Data Types 1.0886 0.005317 0.27182 1	State Attributes		Tune	\Box <u>Compension</u> $P + I \cdot T_s$	$\frac{1}{z-1} + D \frac{N}{1+N \cdot T_s}$	$\frac{1}{z-1}$
Initial condi	tions							
Source:	internal							-
Integrator:	0							
Filter:	0							
0					ОК	Cancel	Help	Apply

Figure 10 – Saisi des paramètres du régulateur

4. Travail à faire

- 1. Réaliser le montage de la figure 1
- 2. Réaliser la structure Simulink figure 9.

3. Faire la simulation pour un correcteur proportionnel $K_p = 40$ et $K_p = 0.4$, que constatez-vous

4. Pour chacun des régulateurs (proportionnel, intégral, proportionnel intégral, *proportionnel dérivé*-Proportionnel Intégrateur Dérivateur), faire le travail suivant :

-Déterminer la fonction de transfert en boucle fermée (retour unitaire)

-Tracer la courbe de la réponse indicielle et de bode

-Déterminer les paramètres suivants : Marge de phase-marge de gain-temps de montée

5. Déduire à partir des courbes précédentes une étude comparative entre les correcteurs

TP4 : Asservissement du processus thermique

1. Objectifs

-Déterminer le modèle du processus thermique

-Implémenter un régulateur numérique dans la carte arduino

2. Présentation de la maquette

La maquette est constituée d'un capteur de température LM35 et une Lampe halogène 12V 35W qui joue le rôle d'un élément chauffant. Le capteur et la lampe sont installés dans une boite en bois avec un couvercle en plexiglas. Cette boite représente le système thermique à commander. La figure suivante schématise la connexion de la carte Arduino UNO avec l'entrée et la sortie du système thermique.



Figure 1 - Branchement du procédé avec la carte Arduino

3. Modélisation du procédé thermique

Le but de cette partie est de déterminer la fonction de transfert continu de notre procédé thermique en boucle ouverte notée H(p).L'entrée du système est la tension u(p) en volts et la sortie est la température T(p) de degré Celsius. Cette étape est constituée de deux parties. La première est assurée par l'environnement Simulink et le package ArduinoIO pour l'envoi et l'acquisition des données. La deuxième partie est assurée par l'outil System identification sous Matlab(voir tp2).



Figure 2 Modèle Simulink pour la détermination de la réponse indicielle

Configurer le bloc to workspace de la manière suivante (figure 3)

Sink Block Parameters: To Workspace1	×						
To Workspace							
Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.							
To log a bus signal, use "Timeseries" save format.							
Parameters							
Variable name:							
cmd							
Limit data points to last:							
inf							
Decimation:							
1							
Sample time (-1 for inherited):							
0.5							
Save format: Array							
Log fixed-point data as a fi object							
	~						
OK Cancel Help Appl	y						

Figure 3 Configurations de bloc to workspace

4. Implémentation de la commande sur la carte arduino

Dans cette partie on va utiliser les fonctions offertes par Arduino pour envoyer (la commande) et acquérir (la température instantanée). L'implémentation du régulateur se fera directement sur la carteArduino.



Figure 4 Schéma synoptique de l'asservissement à l'implémenter

4.1 Le correcteur à retard de phase

Le correcteur à retard de phase est une forme approchée du correcteur PI. Il réalise une action intégrale (augmentation du gain en basses fréquences) sans introduire d'intégrateur. Il est défini par la fonction de transfert suivante

$$FT = K \frac{1 + Tp}{1 + bTp} \quad b > 1$$

Le retard de phase maxi $\Delta \phi$ se produit à $\omega = \omega_a = \frac{1}{T\sqrt{b}}$.

Retard de phase nul pour $\omega > \omega_b = \frac{10}{T\sqrt{b}}$, ω_b est choisi égal à 10 ω_a



Algorithme de réglage

1. On détermine d'abord b :

Soit H0 le gain statique de la BONC.

Soit *KH*0 le gain statique de la BOC (gain statique désiré correspondant à une précision statique requise)

On a

$$20\log(b) = |KH_0|_{db} - |H_0|_{db} = 20\log(K) \to b = K$$

Le gain K du correcteur est choisi égal à b afin que les fréquences $\omega_b = \frac{10}{T_a/b}$ ne soient pas

affectées.

float somme_erreur=0;

2. On détermine *T* tel que à $\omega_b = \frac{10}{T\sqrt{b}}$ ne se produit plus de retard de phase

4.2 Implémentation de régulateur L'implémentation du régulateur retard de phase sur la carte

Le code suivant permet d'implémenter le régulateur de retard de phase sur la carte Arduino. float delta_erreur=0; float delta_commande=0;

```
float ep,up,temp,u,e,cmd;
int consigne=35;
void setup() {
Serial.begin(9600);
}
void loop() {
temp=analogRead(A0);
temp=(temp*500)/1023;
e=consigne-temp;
delta_commande= u-up/0.5
delta_erreur=(e-ep)/0.5;
somme erreur=somme erreur+e*0.5;
cmd=K*e+K*T*somme_erreur+Kd*delta_erreur-b*T* delta_commande
analogWrite(6,cmd*(255/12));
ep=e;
up=u;
```

delay(500);//période d'échantillonnage

Serial.write(analogRead(0));//envoi de la donnée sur le port série}

Il suffit d'exploiter la bibliothèque Instrument Control Toolbox pour la lecture de la température instantanée.



Figure 5 Acquisition de la température sous instrument Control Toolbox

5. Travail à faire

- 1. Réaliser le montage de la figure 1
- 2. Réaliser la structure Simulink figure 2.
- 3. Déterminer la fonction de transfert en boucle ouverte
- 4. Tracer la réponse indicielle et le diagramme de bode

5. En utilisant PID tuning tracer la réponse indicielle et déterminer le temp de montée pour un correcteur proportionnelle dérivé

6. On considère un correcteur à retard de phase

-Déterminer les valeurs de b et T afin d'avoir une erreur statique de 2% et une marge de phase 60° (échelon est unitaire)

- Implémenter le régulateur dans la carte arduino pour les valeurs trouvées et réaliser la structure Simulink figure 5(déduire le code de régulateur à retard de phase à partir de code du régulateur pid donné)

-Vérifier les performances désirées