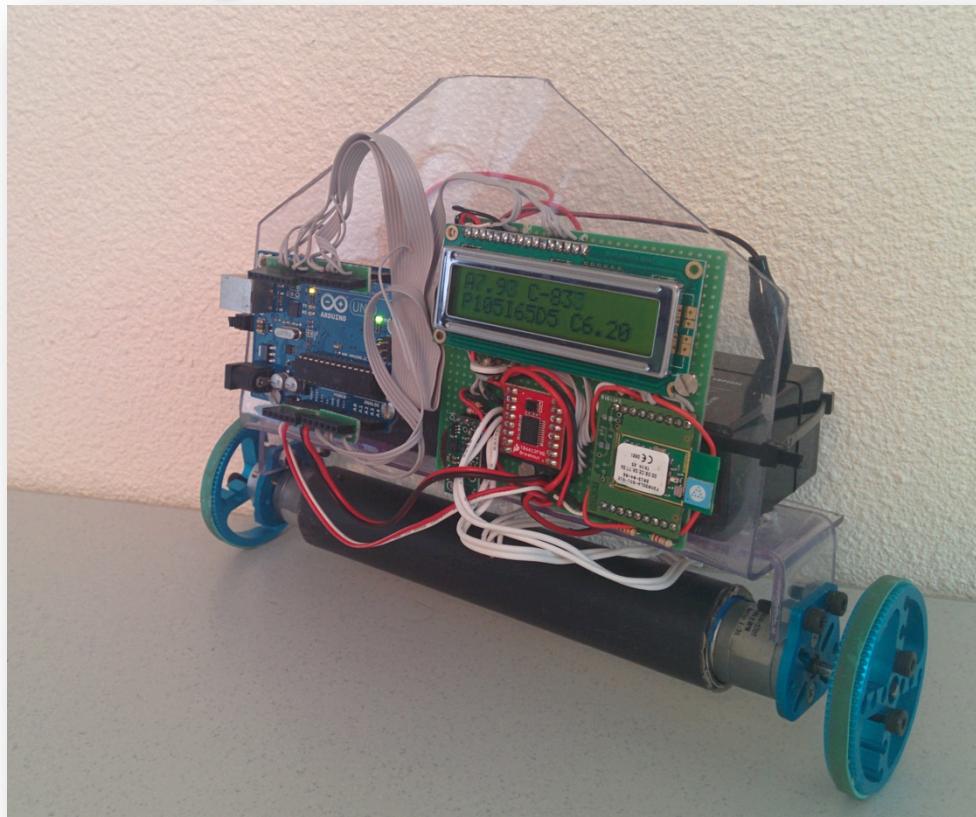




Asservissement en position d'équilibre instable



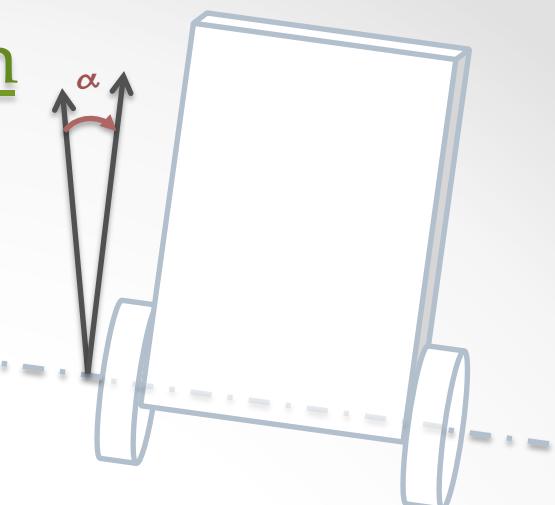
Conception de la commande d'un robot pendulaire

- I. Présentation de la stratégie de résolution
- II. Acquisition et filtrage de l'inclinaison du pendule
Transition : dimensionnement de la partie opérative
- III. Conception de l'asservissement
- IV. Caractérisation de la stabilité

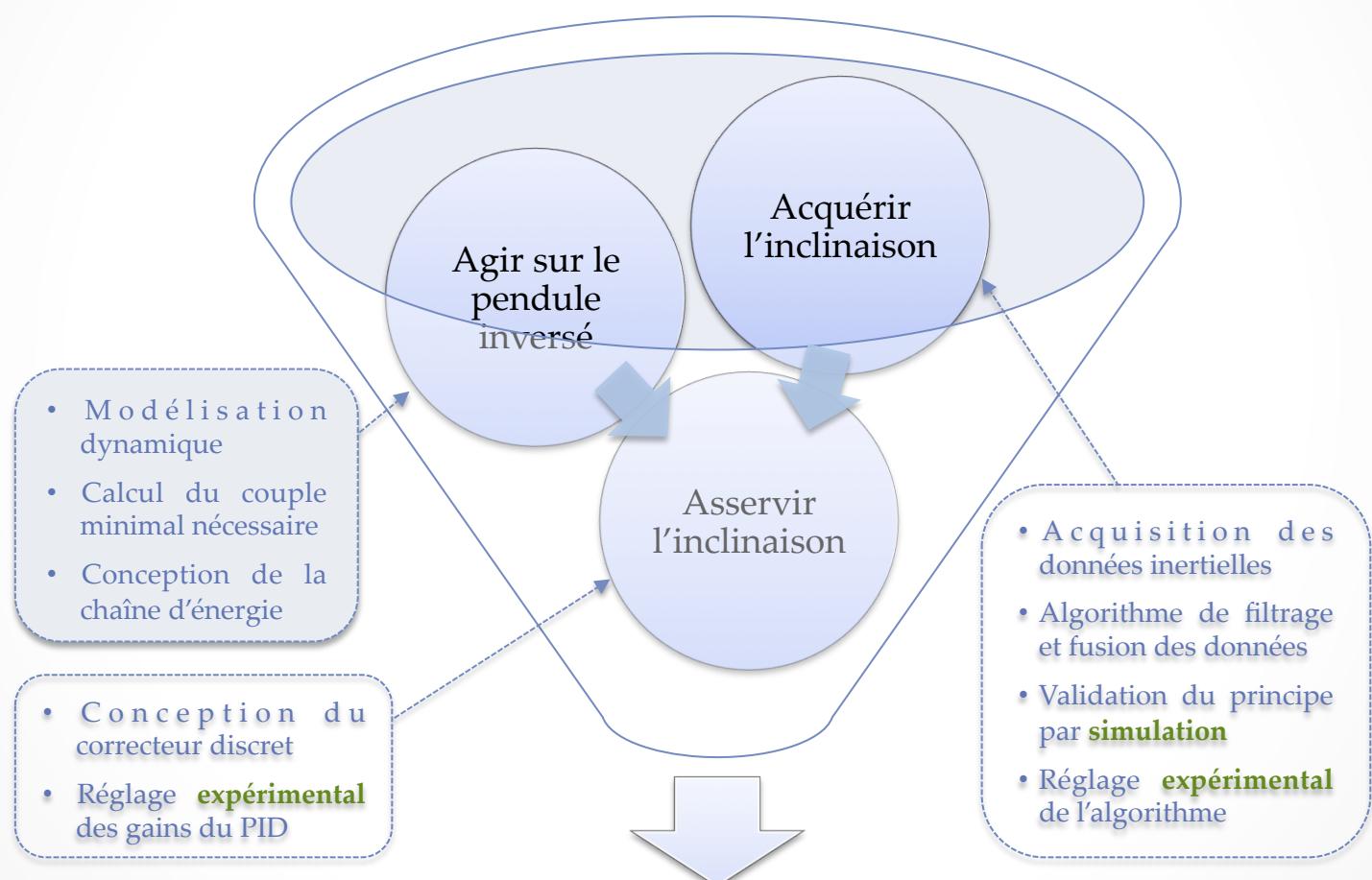
I. Stratégie de résolution

Structure de base : Pendule inversé

Contraintes : autonome et autosuffisant



Comment maintenir un robot pendulaire à la verticale ?



Asservir l'inclinaison du robot pendulaire



Validation : évaluation **expérimentale** de la stabilité

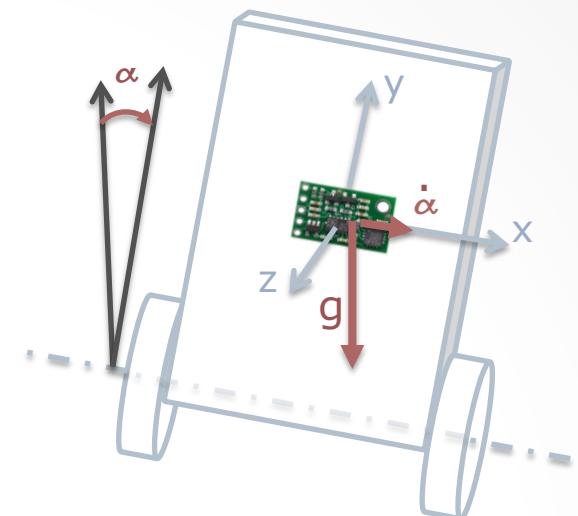
II. Acquisition et filtrage de l'inclinaison

1. Capteurs utilisés

Capteurs inertIELS : accéléromètres et gyromètre

Deux sources d'information

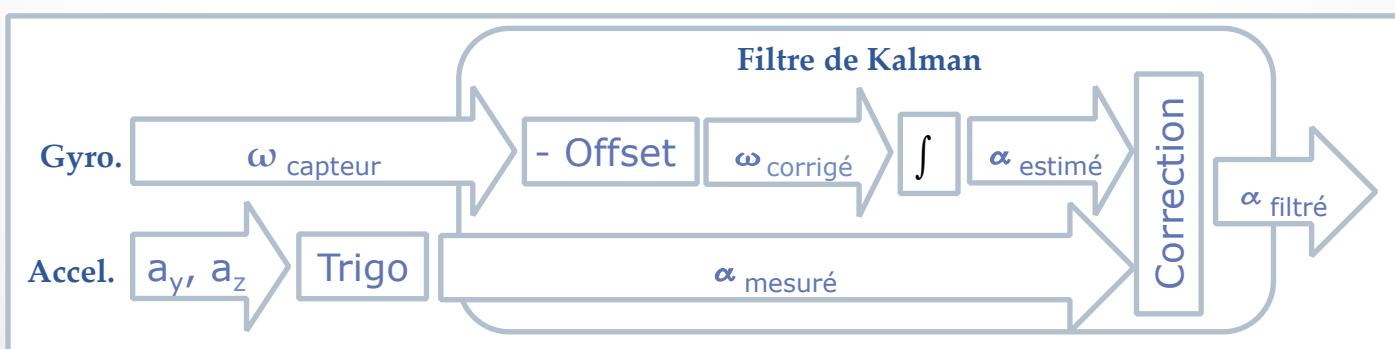
Accélération de pesanteur + Trigonométrie	Taux de giration + Intégration
	
Accélérations parasites	Divergence
Fiable au repos et à long terme (BF)	Fiable à court terme (HF)



Capteurs complémentaires → Fusion de données

2. Principe du filtre de Kalman (Annexe 1)

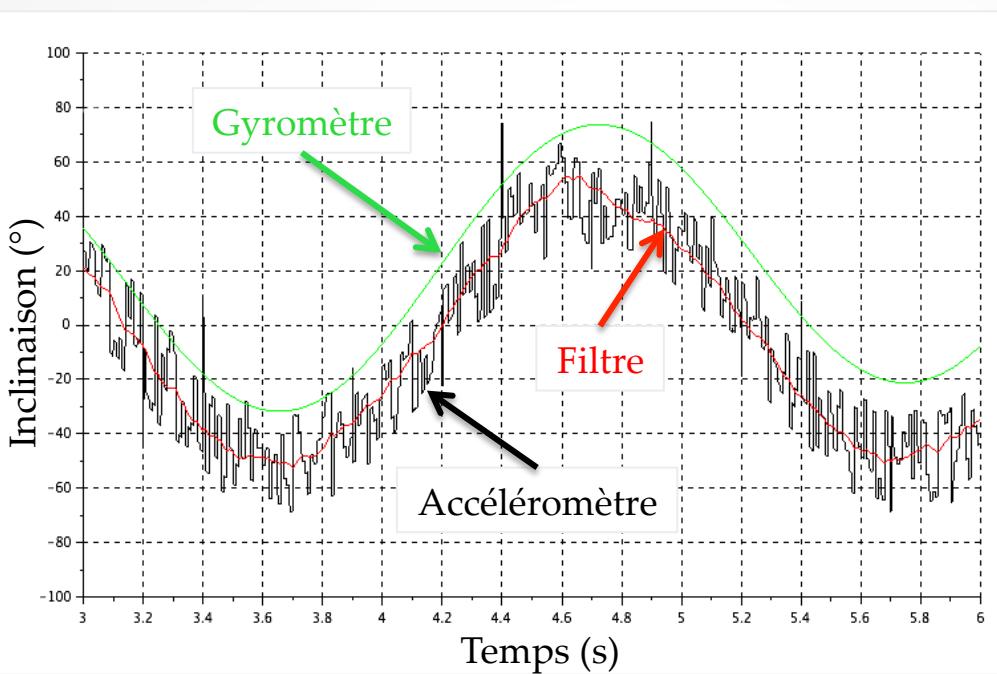
- Filtre récursif (à mémoire)
- Annule les biais et estime l'inclinaison



II. Acquisition et filtrage de l'inclinaison

3. Simulation informatique (Annexe 2)

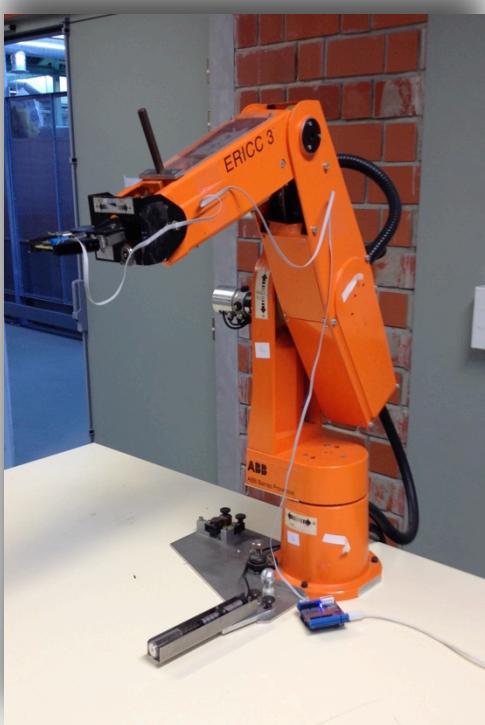
SciLab – Xcos :



Objectifs :

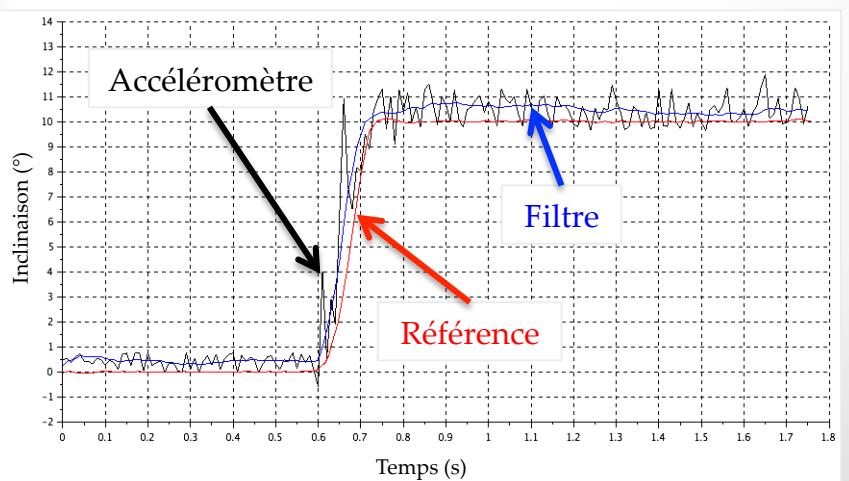
- Valider le principe
- Prérégler les gains du filtre

4. Validation expérimentale (Annexe 3)



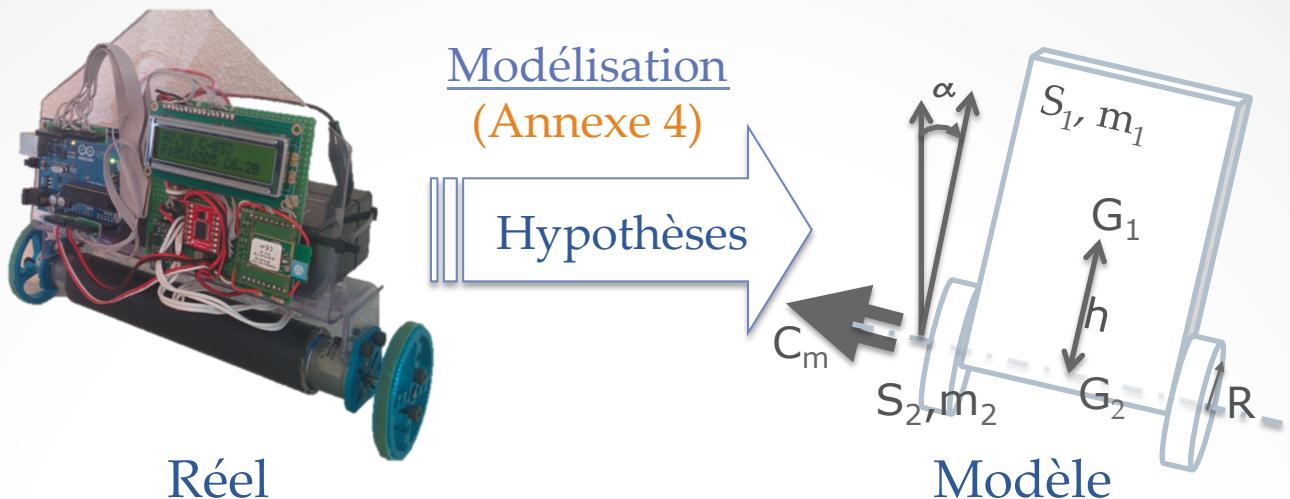
Scilab – Xcos
+ interface d'acquisition Arduino - I2C

Objectif :
Réglage du filtre (variance des capteurs)



Transition : Dimensionnement de la P.O.

1. Dimensionnement des moteurs



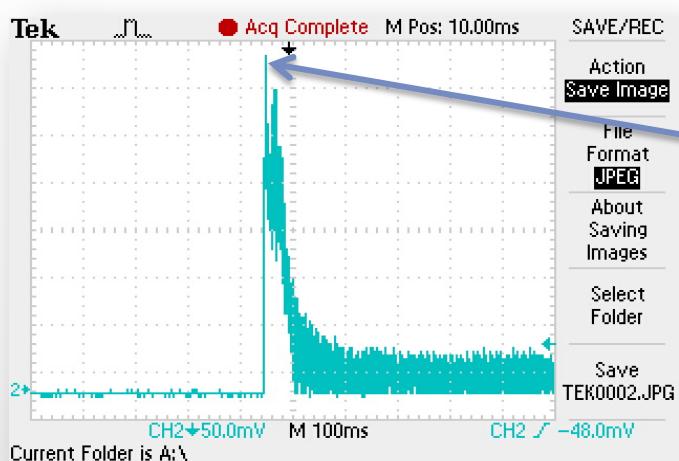
$$\begin{cases} \ddot{\alpha}(A_1 + m_1 h^2) + \ddot{x}(m_1 h \cos \alpha) - m_1 h g \sin \alpha + 2C_m = 0 \\ \ddot{\alpha}(m_1 h \cos \alpha) + \ddot{x}(m_1 + m_2 + \frac{A_2}{R^2}) - m_1 h \dot{\alpha}^2 \sin \alpha - \frac{2C_m}{R} = 0 \end{cases}$$

Critère du couple minimal
(Annexe 5)

Calcul critique → Couple minimal
→ Choix moteurs



2. Dimensionnement de la commande



Courant crête moteur



Choix du hacheur

III. Conception de l'asservissement

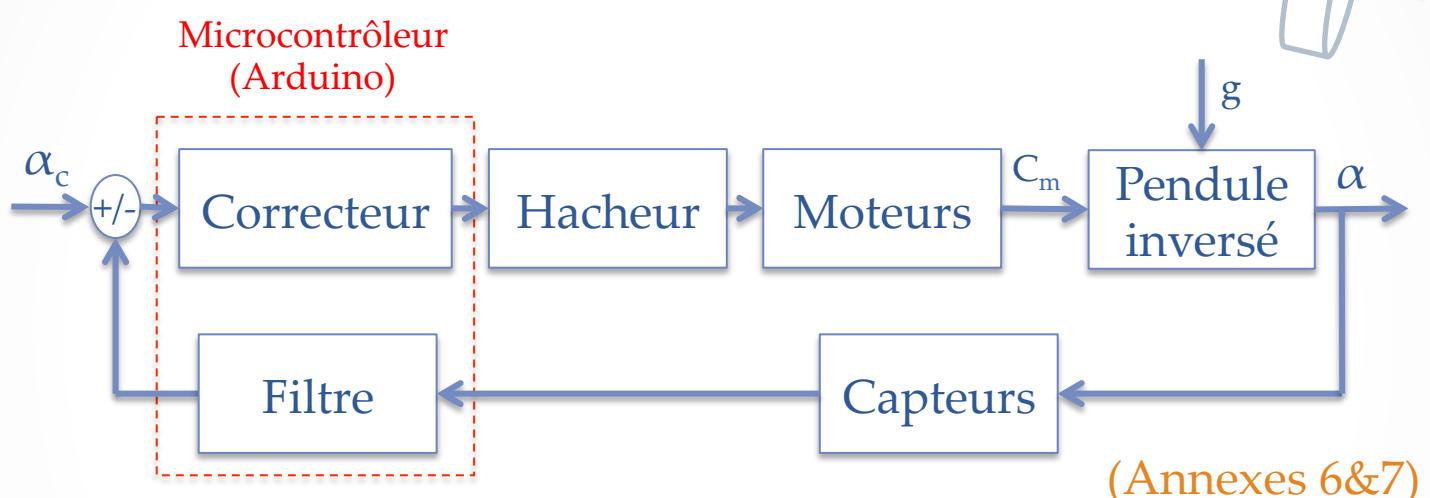
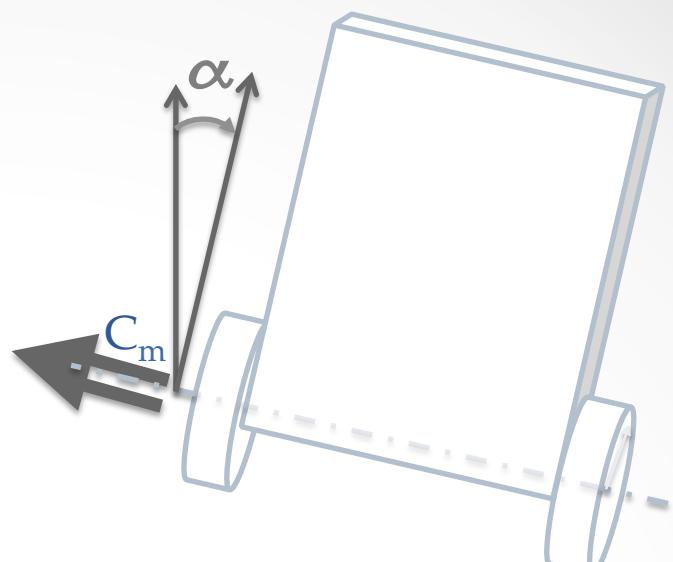
1. Action du correcteur

Principe :

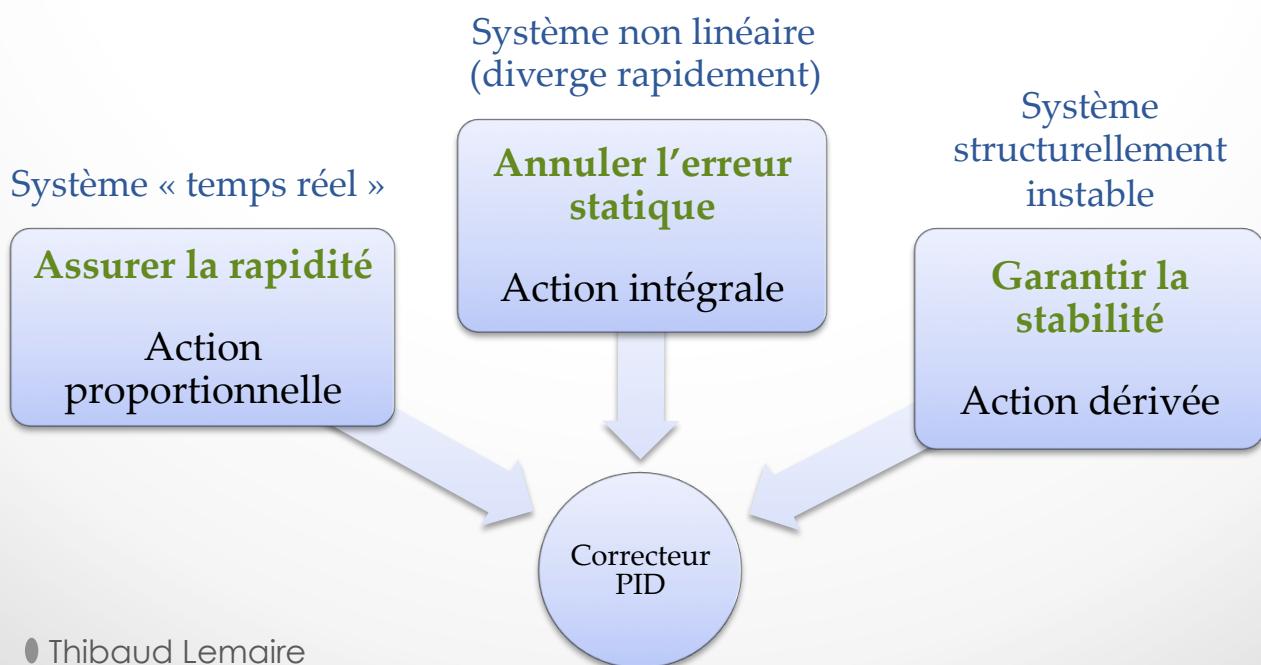
Commande des moteurs en tension

Conséquence :

Couple de maintien fonction de la dynamique du pendule



2. Choix du correcteur



III. Conception de l'asservissement

3. Implémentation du correcteur (Annexe 8&9)

$I_n = I_{n-1} + \varepsilon_n \cdot T_{ech}$ Action intégrale : méthode des rectangles

$D_n = \frac{\varepsilon_n - \varepsilon_{n-1}}{T_{ech}}$ Action dérivée : méthode d'Euler

$$Correction_n = K_p \cdot \varepsilon_n + K_i \cdot I_n + K_d \cdot D_n \quad f_{\text{éch}} = 300 \text{ Hz}$$

```
// Fonction majPid, met à jour la correction PID
void pid::majPid(double mesure, double dt)
{
    double erreur = (consigne - mesure);
    double variationErreurs = int((erreur - erreurPrecedente)*100); // Partie entière
    sommeErreurs = contraindre(sommeErreurs + erreur * dt, 1, -1);
    termeP = KP * erreur;
    termeI = KI * sommeErreurs * 10;
    termeD = KD * variationErreurs / dt /1000; // /100 rétabli l'échelle
    correction = termeP + termeI + termeD;
    erreurPrecedente = erreur;
}
```

4. Réglage du correcteur

Protocole : réglage « à la volée »

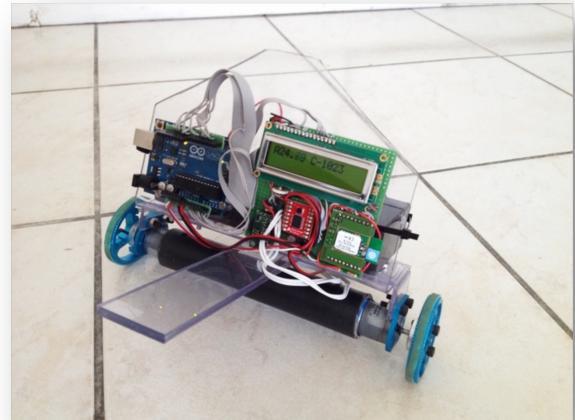
Annulation de tous les gains

Réglage grossier de Kp
(permettre le basculement, rapidité)

Réglage de Ki
(retour à l'équilibre, annuler l'erreur)

Réglage de Kd
(assurer le maintien en équilibre, stabilité)

Ajustement de Kd
(limiter les oscillations)



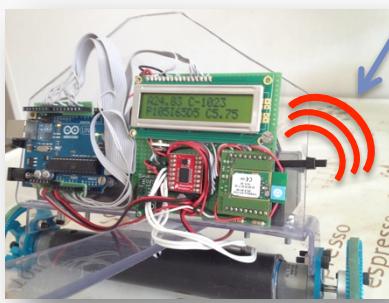
Robot avec barre antichute



IV. Caractérisation de la stabilité

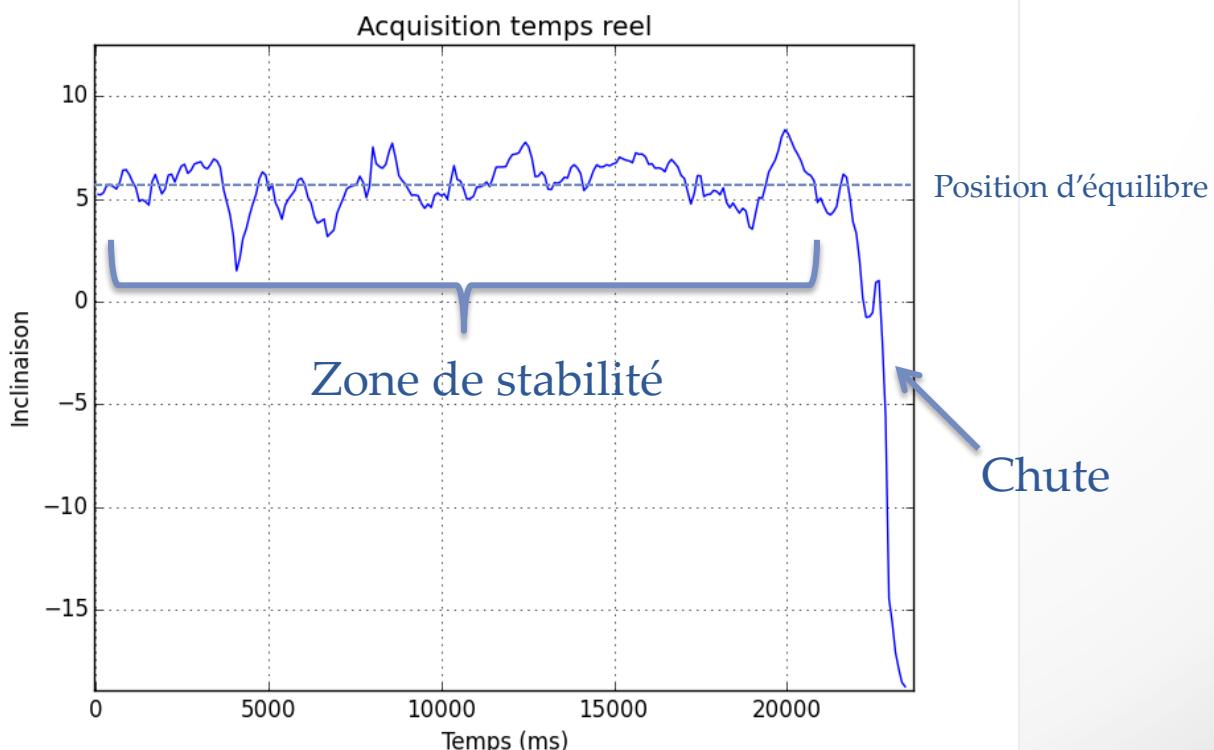
1. Dispositif expérimental

Liaison série
Via Bluetooth



```
Spyder (Python 2.7)
Éditeur - /Users/thibaud/Documents/Prepa/PT*/Divers/Liveplot/liveplotv1-2.py sim.py
16
17 def addValue(temps, val):
18     tps.append(val)
19     temps.append(temps)
20
21 def communication():
22     global ser
23     ser.flush()
24     donnees = [0]
25     print("Début du thread d'acquisition")
26     while donnees[0] <> "F":
27         try:
28             ligne = ser.readline()
29             donnees = ligne.split(" ")
30             if donnees[0] == "E":
31                 addValue(float(donnees[1]), float(donnees[2]))
32             if donnees[0] == "S":
33                 pass
34             time.sleep(0.001)
35         except ValueError:
36             pass
37     print("Fin du thread d'acquisition")
38
39
40 def connect():
41     global ser
42     print("Connexion...")
43     ser = serial.Serial('/dev/tty.Free2moveU-SerialPort')
44     print("Connecté !")
45     ser.flush()
46
47 def disconnect():
48     global ser
49     ser.close()
50     print("Deconnecté !")
51
52 def affichage():
53     plt.plot(tps, data)
54     plt.grid()
55
56 Droits d'accès : RW Fins de ligne : LF Encodage : UTF-8
Figure 1
Acquisition temps réel
Inclinaison
25
20
15
10
5
0
-5
-10
-15
-20
-25
2000 4000 6000 8000 10000 12000
Temps (ms)
zoom rect, x=3172 y=24.8768
Ligne : 34 Colonne : 1 Mémoire :
```

2. Résultats



Plan détaillé

- I. Présentation de la stratégie de résolution
- II. Acquisition et filtrage de l'inclinaison du pendule
 - 1) Capteurs utilisés
 - 2) Principe du filtre de Kalman (Annexe 1)
 - 3) Simulation numérique du filtre (Annexe 2)
 - 4) Réglage expérimental du filtre (Annexe 3)

Transition : dimensionnement de la partie opérative (Annexes 4&5)

- III. Conception de l'asservissement
 - 1) Action du correcteur (Annexes 6&7)
 - 2) Choix du correcteur
 - 3) Implémentation du correcteur (Annexes 8&9)
 - 4) Réglage expérimental du correcteur
- IV. Caractérisation de la stabilité

Démarche

- I. Acquérir l'inclinaison du pendule inversé
- II. Agir sur le pendule inversé (partie opérative)

Puis, grâce à ces ressources :

- III. Asservir l'inclinaison

Enfin, une fois le système opérationnel

- IV. Caractériser la stabilité

Annexe 1 : Filtre de Kalman

Estimation de l'état, *a priori* :

$$\hat{x}_{k|k-1} = \begin{pmatrix} 1 & \Delta T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta \\ \dot{\theta}_b \end{pmatrix}_{k-1|k-1} + \begin{pmatrix} \Delta T \\ 0 \end{pmatrix} \dot{\theta}_k \quad \text{Mesure gyro}$$

$$\hat{x}_{k|k-1} = \begin{pmatrix} \theta_{k-1} + \Delta T(\dot{\theta}_k - \dot{\theta}_{b,k-1}) \\ \dot{\theta}_{b,k-1} \end{pmatrix} \quad \boxed{\hat{x}_{k|k-1} = \begin{pmatrix} \theta \\ \dot{\theta}_b \end{pmatrix}_{k|k-1}}$$

Calcul de la matrice de covariance de l'erreur, *a priori* :

$$\mathbf{Q}_k = \begin{pmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{pmatrix} (\Delta T)^k \quad \text{Variances accel/gyro}$$

$$\mathbf{F} = \begin{pmatrix} 1 & -\Delta T \\ 0 & 1 \end{pmatrix} \quad \text{Matrice de transition d'état}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F} \mathbf{P}_{k-1|k-1} \mathbf{F}^T + \mathbf{Q}_k$$

Calcul de l'innovation

$$\tilde{\mathbf{y}}_k = \theta_{m,k} - \theta_{k|k-1} \quad \text{Ecart entre la prédiction et la mesure}$$

Calcul de la covariance de l'innovation

$$R = \text{var}(\theta_{m,bruit}) \quad \text{Variance du bruit de mesure}$$

$$\mathbf{S}_k = P_{00,k|k-1} + R$$

Calcul des gains de Kalman

$$\mathbf{K}_k = \begin{pmatrix} K_0 \\ K_1 \end{pmatrix}_k = \frac{1}{\mathbf{S}_k} \begin{pmatrix} P_{00} \\ P_{10} \end{pmatrix}_{k|k-1} \quad \text{À quelle source faire confiance ?}$$

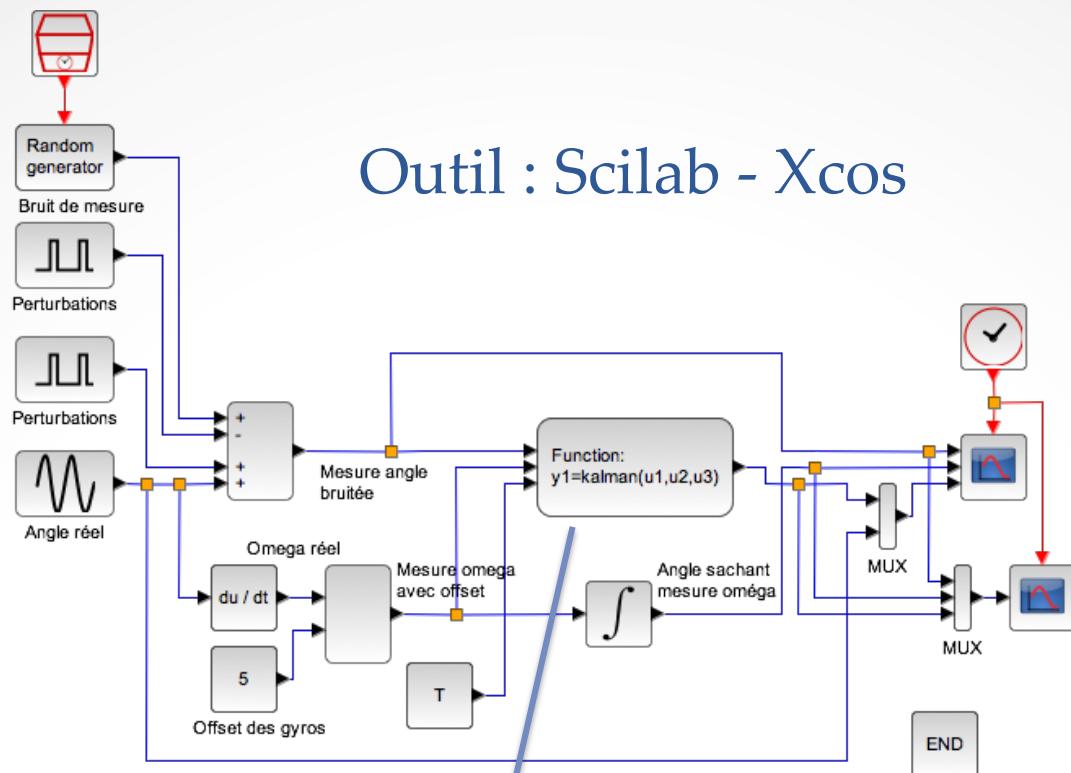
Mise à jour du modèle

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$$

Mise à jour de la covariance de l'erreur

$$\mathbf{P}_{k|k} = (\mathbf{I}_2 - \mathbf{K}_k \begin{pmatrix} 1 & 0 \end{pmatrix}) \mathbf{P}_{k|k-1}$$

Annexe 2 : Simulation du filtre de Kalman



```

// Fonction kalman
// Cette fonction effectue un filtrage récursif de Kalman sur les données interties
// en provenance des capteurs
function teta=kalman(nouvel_angle, nouveau_taux, dt)
    // Définition des variables globales
    global biais           // Biais actuel des gyromètres
    global angle            // Angle tête actuel du modèle
    global P                // Matrice de covariance de l'erreur

    // Définition des constantes :
    Q_angle = 0.001;        // Variance de l'angle 0.001
    Q_biais = 0.003;        // Variance du biais 0.003
    R_angle = 0.03;         // Variance de la mesure de l'angle 0.03

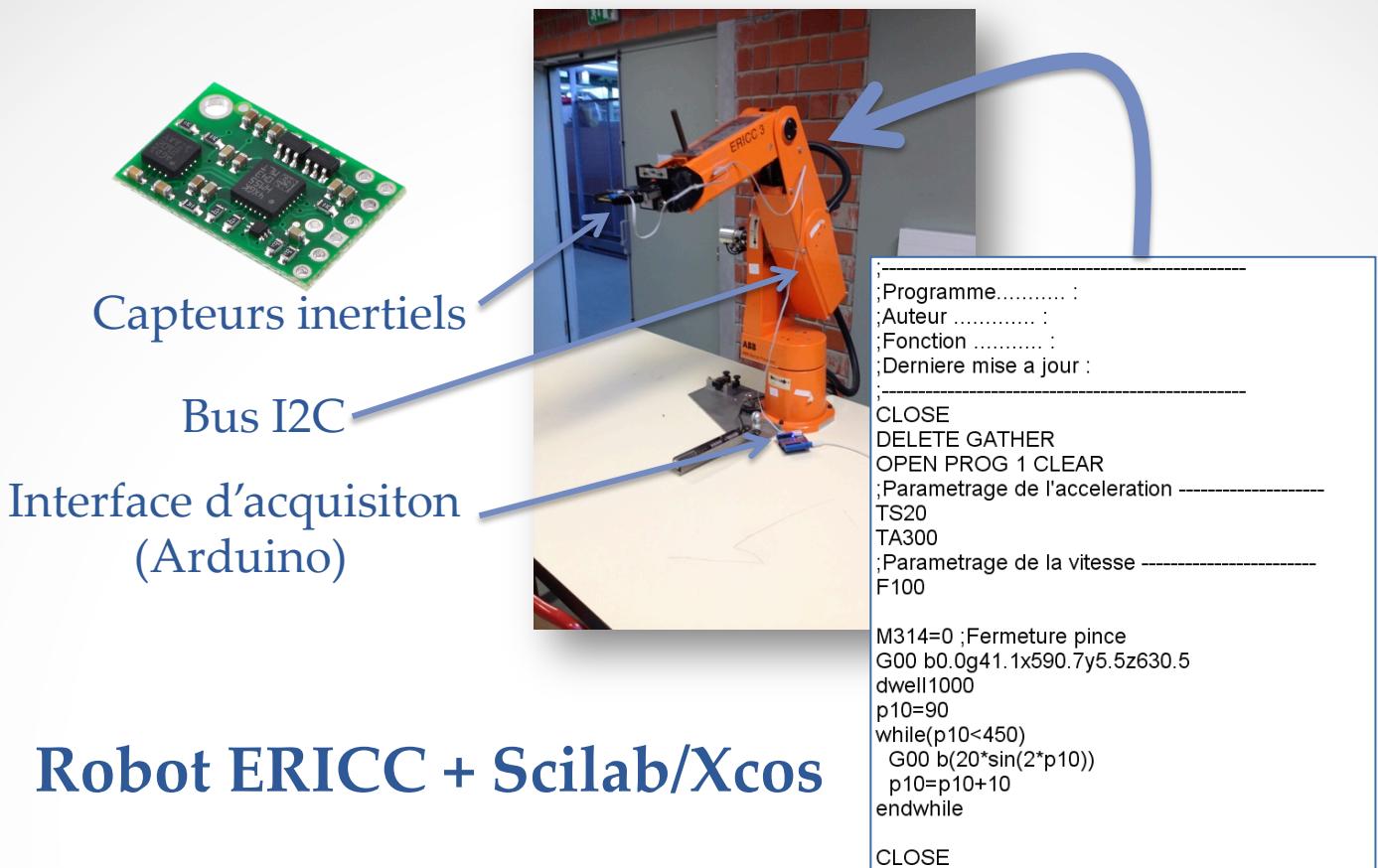
    // Phase de prédiction :
    taux = nouveau_taux - biais; // Correction du taux de gyration
    angle = angle + dt * taux; // Intégration du taux de gyration
    // Calcul de l'estimation de la covariance de l'erreur :
    P(1,1) = P(1,1) + (dt * P(2,2) - P(1,2) - P(2,1) + Q_angle));
    P(1,2) = P(1,2) - (dt * P(2,2));
    P(2,1) = P(2,1) - (dt * P(2,2));
    P(2,2) = P(2,2) + (Q_biais * dt);

    // Phase de mise à jour :
    y = nouvel_angle - angle; // Calcul de l'innovation
    S = P(1,1) + R_angle; // Calcul de la covariance de l'innovation
    // Calcul des gains de Kalman :
    K0 = P(1,1)/S; // Calcul du gain de Kalman de l'angle
    K1 = P(2,1)/S; // Calcul du gain de Kalman du biais
    // Mise à jour du modèle :
    angle = angle + K0 * y; // Mise à jour de teta
    biais = biais + K1 * y; // Mise à jour du biais
    // Mise à jour de la covariance de l'erreur :
    P(1,1) = P(1,1) - (K0 * P(1,1));
    P(1,2) = P(1,2) - (K0 * P(1,2));
    P(2,1) = P(2,1) - (K1 * P(1,1));
    P(2,2) = P(2,2) - (K1 * P(1,2));

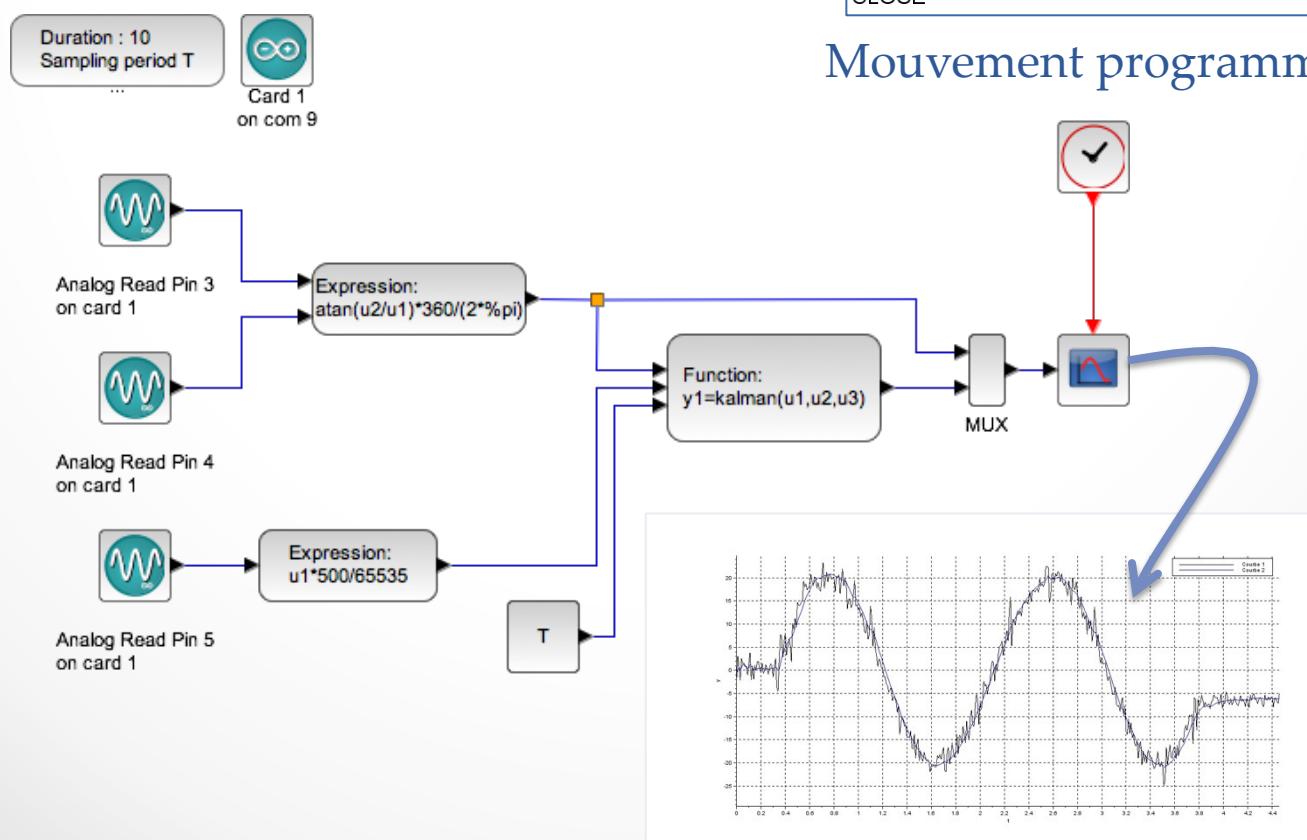
    // Retour de l'angle du modèle :
    teta = angle;
endfunction

```

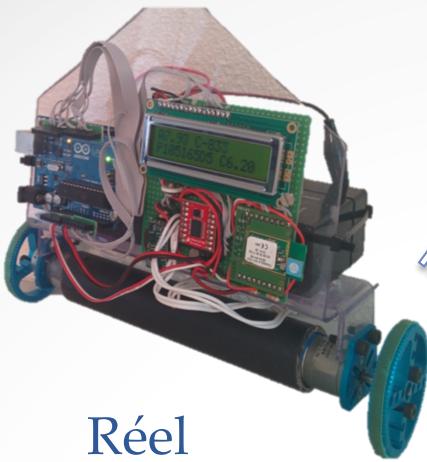
Annexe 3 : Réglage expérimental du filtre



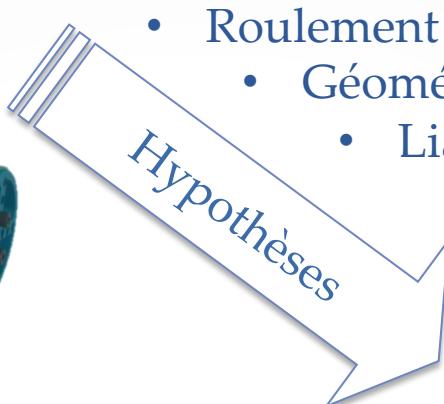
Robot ERICC + Scilab/Xcos



Annexe 4 : Modélisation dynamique



Réel

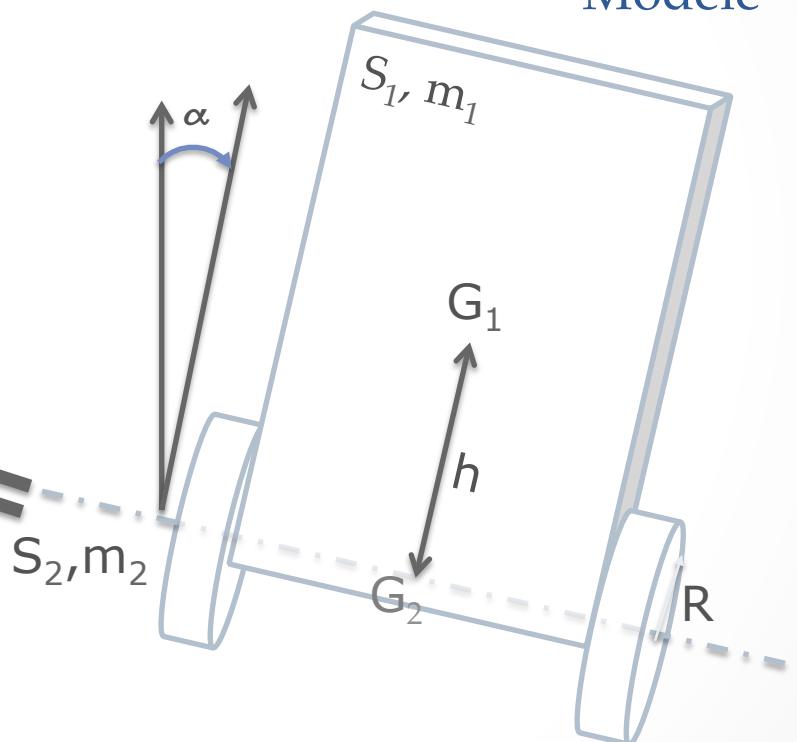


- Roulement sans glissement
- Géométries simples
- Liaisons parfaites

Modèle

$$I_1 = \begin{bmatrix} A_1 & 0 & 0 \\ 0 & B_1 & 0 \\ 0 & 0 & C_1 \end{bmatrix}_{G_1}$$

$$I_2 = \begin{bmatrix} A_2 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & B_2 \end{bmatrix}_{G_2}$$



PFD appliqué à S_1 et $\{S_1 + S_2\}$:

$$\begin{cases} \ddot{\alpha}(A_1 + m_1 h^2) + \ddot{x}(m_1 h \cos \alpha) - m_1 h g \sin \alpha + 2C_m = 0 \\ \ddot{\alpha}(m_1 h \cos \alpha) + \ddot{x}(m_1 + m_2 + \frac{A_2}{R^2}) - m_1 h \dot{\alpha}^2 \sin \alpha - \frac{2C_m}{R} = 0 \end{cases}$$

Annexe 5 : Dimensionnement des moteurs

Détermination du couple minimal → Calcul critique

$$\begin{cases} \ddot{\alpha}(A_1 + m_1 h^2) + \ddot{x}(m_1 h \cos \alpha) - m_1 h g \sin \alpha + 2C_m = 0 \\ \ddot{\alpha}(m_1 h \cos \alpha) + \ddot{x}(m_1 + m_2 + \frac{A_2}{R^2}) - m_1 h \dot{\alpha}^2 \sin \alpha - \frac{2C_m}{R} = 0 \end{cases}$$

Paramètres fixés

- Pas de correction d'inclinaison
- Départ à l'arrêt

$$\begin{cases} C_m = 0 \\ \ddot{x} = 0 \end{cases}$$

$$\begin{cases} \ddot{\alpha}(A_1 + m_1 h^2) = m_1 h g \sin \alpha \\ \ddot{\alpha}(m_1 h \cos \alpha) - \dot{\alpha}^2(m_1 h \sin \alpha) = 0 \end{cases}$$

Pour α fixé ($à 5^\circ$ puis 10°) :

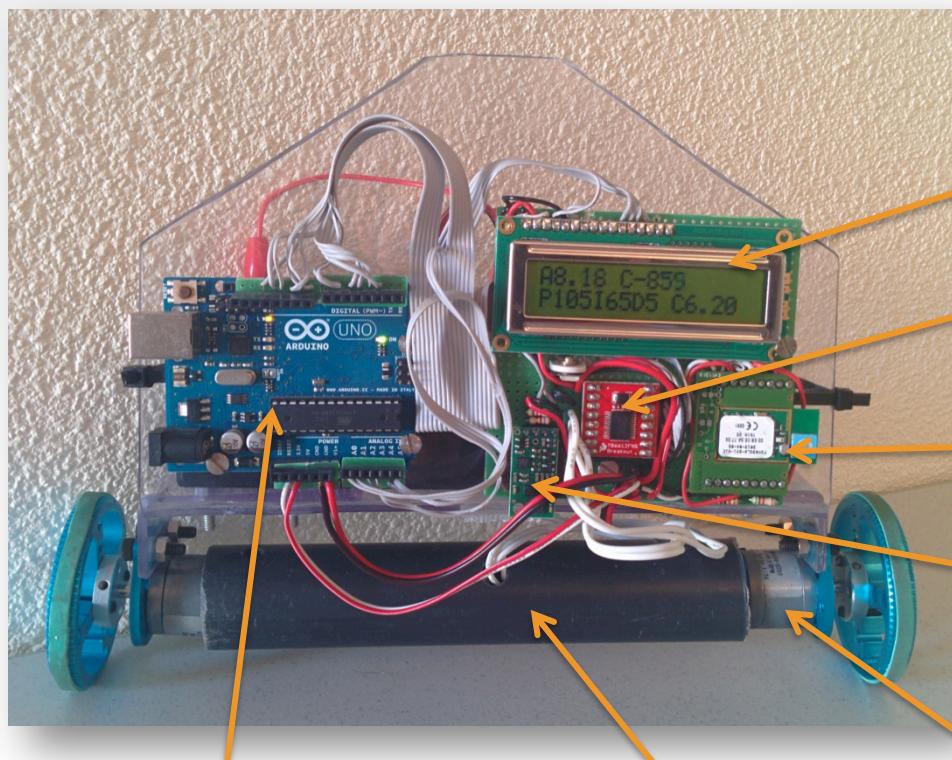
Résolution du système d'inconnues $\ddot{\alpha}, \dot{\alpha}^2$

Détermination du couple critique :
Résolution du système initial en C_m, \ddot{x}



Pour $\alpha = 5^\circ, C_{m, \min} = 5.10^{-3} \text{ N.m}$

Annexe 6 : Conception de la maquette



Carte Arduino

Barre stabilisatrice

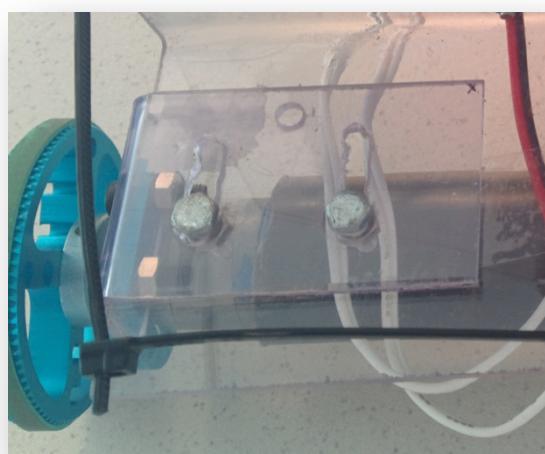
Afficheur LCD

Driver moteurs

Module Bluetooth

Capteurs inertIELS

Moteur



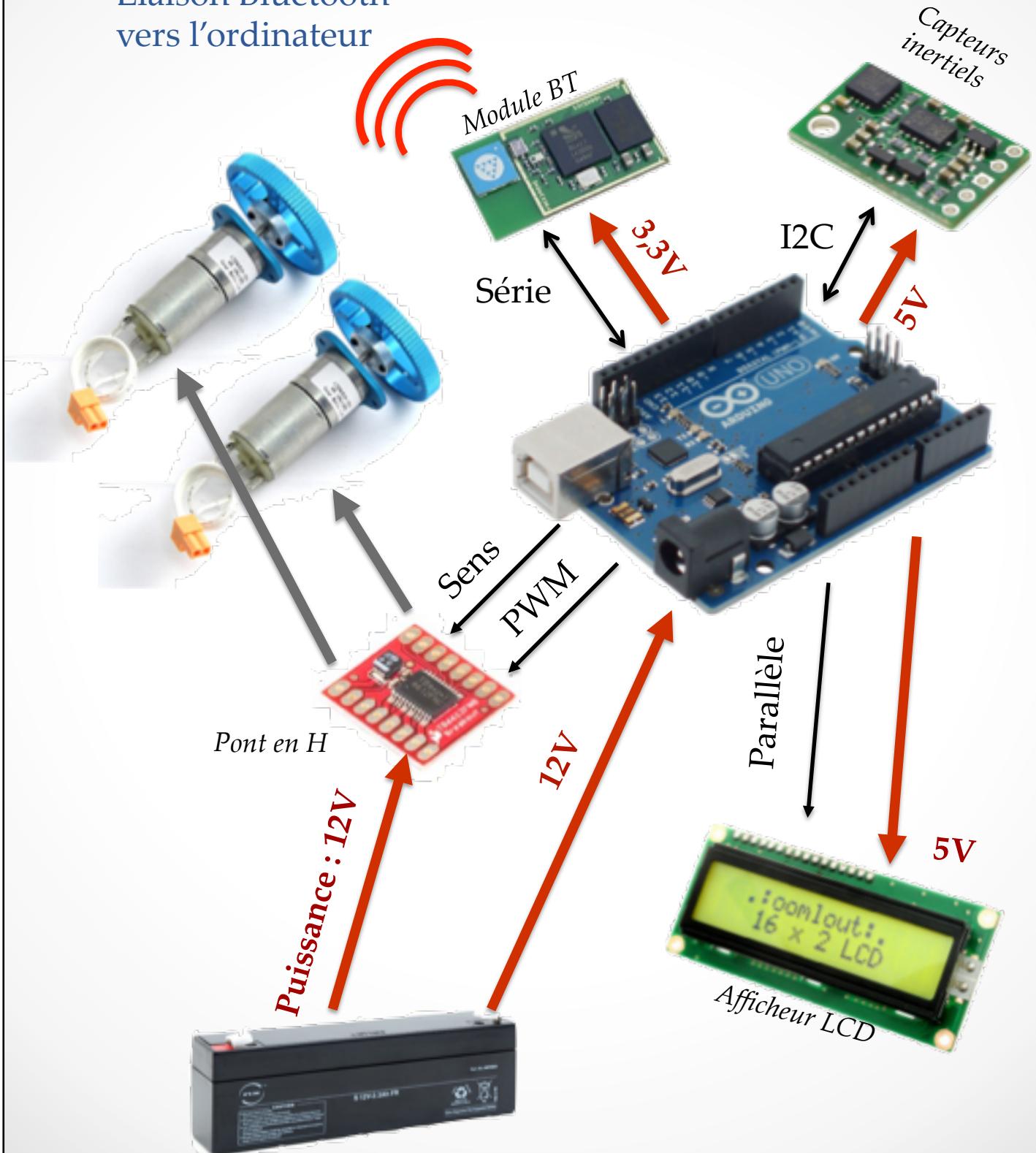
Réglage du centre d'inertie
Trous oblongs



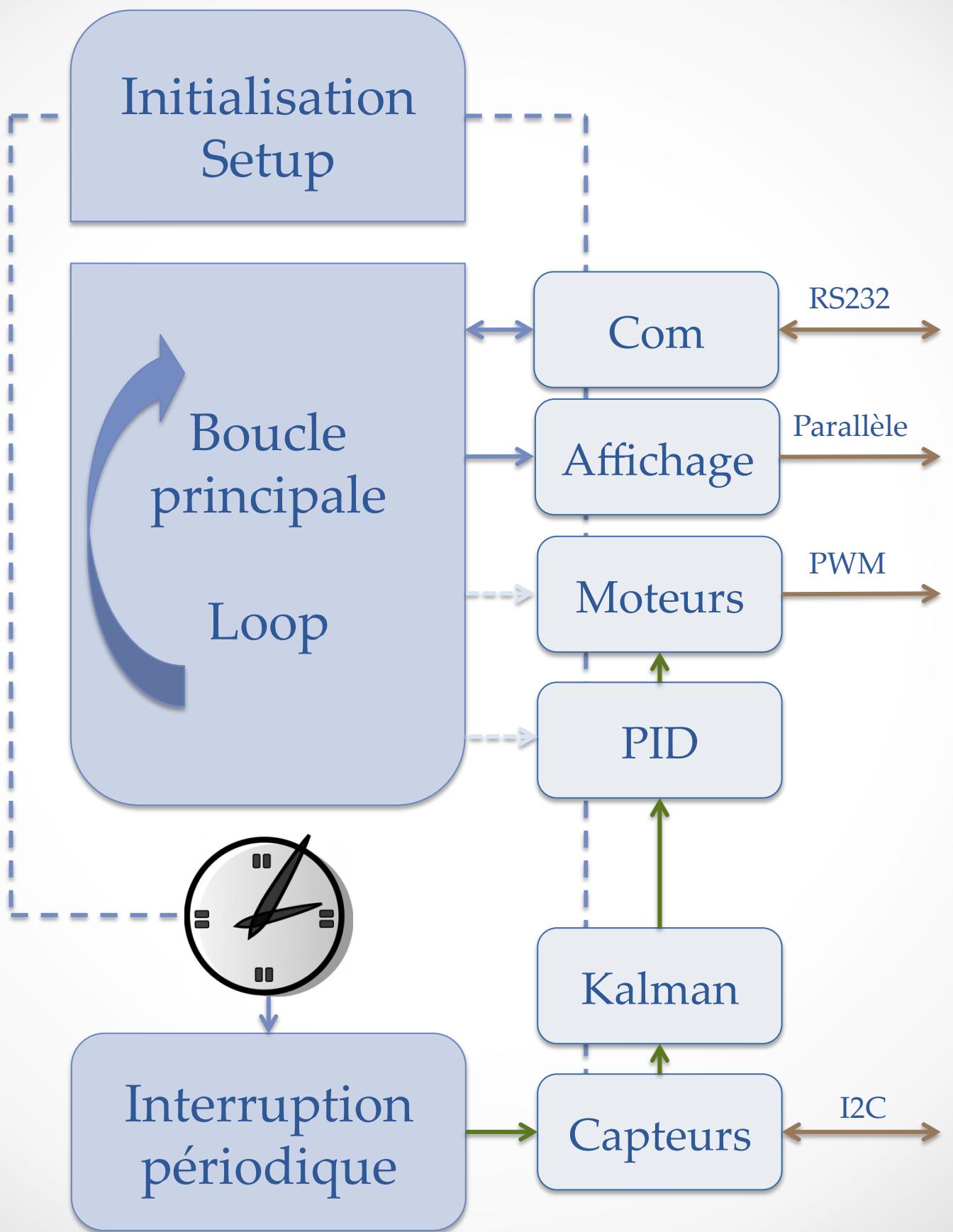
Augmentation de l'adhérence
Elastiques

Annexe 7 : Architecture matérielle

Liaison Bluetooth
vers l'ordinateur



Annexe 8 : Architecture logicielle du robot



Annexe 9 : Conception du soft embarqué : Xcode

The screenshot shows the Xcode interface with the project navigation bar at the top. The project name is "Robobal". Below it, the file structure is displayed:

- Robobal (selected)
- ↳ 5 targets... X SDK 10.10 M
- ↳ Robobal
- ↳ MsTimer2.cpp
- ↳ MsTimer2.h
- ↳ robot.ino
- ↳ capteur.cpp
- ↳ capteur.h
- ↳ commu...ion.cpp
- ↳ communication.h
- ↳ kalman.cpp
- ↳ kalman.h
- ↳ main.cpp
- ↳ moteurs.cpp
- ↳ moteurs.h
- ↳ pid.cpp
- ↳ pid.h
- ↳ Makefile
- ↳ ReadMe.txt
- ↳ Sketchbook
- ↳ Configurations
- ↳ Makefiles
- ↳ About
- ↳ Utilities
- ↳ Products

The code editor shows the following C++ code for the "kalman.cpp" file:

```
#include <Arduino.h>
#include <math.h>
#include "kalman.h"

// Developed with embedXcode
// http://embedXcode.weebly.com

// Project      robot
// Created by   Thibaud Lemaire, 07/05/2014 12:06
//             Thibaud LEWAIRE
// Copyright    © Thibaud Lemaire, 2014
// License      license
// See          kalman.h and ReadMe.txt for references

// Library header
#include "kalman.h"

// Constructeur du filtre
kalman::kalman()
{
    // Valeurs par défaut des variances
    Qangle = 0.001; // 0.001
    Qbiais = 0.003; // 0.003
    Rmesure = 0.03; // 0.03
}

// Initialisation du modèle
angle = 0;
biais = 0;

// En supposant l'état initial connu, la matrice de
// covariance de l'erreur est nulle
P[0][0] = 0;
P[0][1] = 0;
P[1][0] = 0;
P[1][1] = 0;

// Mise à jour du modèle
void kalman::acq(double angleMesure, double omegaMesure
, double dt) {
    // Déclaration des variables
    double S; // Estimation de l'erreur
    double K[2]; // Gains de Kalman
}
```

Below the code editor, the status bar shows "Robobal | Build All: Succeeded | 02/04/2015 at 14:26".