Laboratory 4
Control of a balancing robot

A Introduction

In this laboratory, you will investigate how to **control a balancing robot**. We will utilize the **Pololu Balboa 32U4 Balancing Robot**[1], pictured in Figure 1. Its control board is compatible with **Arduino**, an open-source platform for prototyping electronic devices.

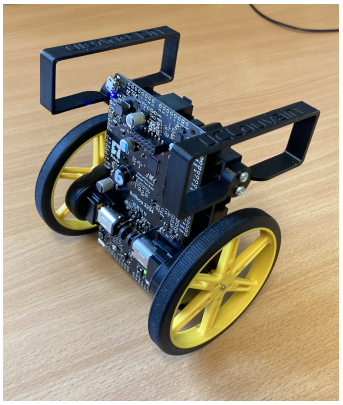

Figure 1: Picture of the Pololu Balboa 32U4 Balancing Robot

The goal of this laboratory covers the most important steps in the design of a control system:

1. **Modeling**: obtain a dynamic model of the robot.
2. **Design**: compute an appropriate control law.
3. **Simulation**: simulate the behavior of the robot and verify the performance of the control law.
4. **Implementation**: see how a controller is practically implemented.

Important

A pre-requisite for this lab, which needs to be completed at home before coming to this lab, is to study in detail and develop (on paper) the Simulink model described in Section D.3. On Moodle, you will find a sample file where an analogous Simulink model has been derived for a similar robot. Use this to prepare.

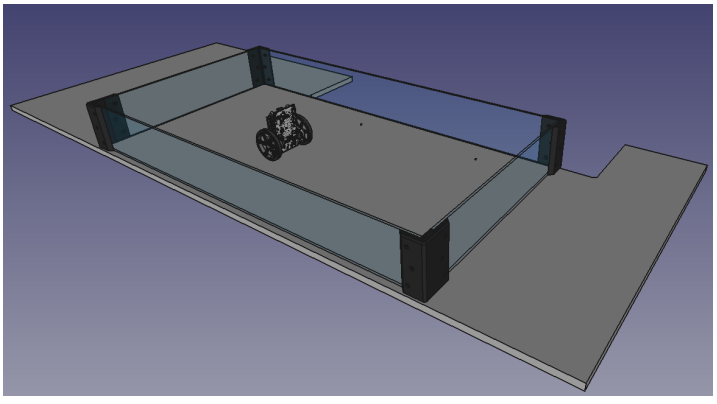

Figure 2: Safe area to test the robot.

[1] See `https://www.pololu.com/docs/0J70` for more documentation.

B Modeling

In this section, we present the derivation of a model for the robot. Notice that the modeling of the mechanical part has been previously studied in TP1, and a video presenting its derivation is also available on Moodle. Moreover, the modeling of the electrical system has been previously discussed in class, and it is available in Section 2 of the course lecture notes.

B.1 Kinematic model of the mechanical system

The cross-section of the kinematic model of the robot and all variables involved are presented in Figure 3. The robot is modeled as inverted pendulum mounted on a moving cart attached to two wheels.

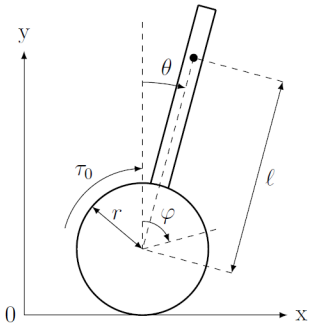

Figure 3: Kinematic model of the robot.

According to Figure 3, the position of the center of mass of the pendulum (or beam) is given by:

$$x_p = x_w + \ell \sin\theta, \tag{1a}$$
$$y_p = y_w + \ell \cos\theta. \tag{1b}$$

The translation velocity of the center of the wheels, denoted by $\dot{x}_w$, is proportional to the angular velocity of the wheels $\dot{\varphi}$:

$$\dot{x}_w = r\dot{\varphi}, \tag{2}$$

and determines the horizontal position of the wheels, denoted by x_w:

$$x_w = x_w(0) + r\varphi. \tag{3}$$

The vertical position of the wheels, denoted by y_w, is constant:

$$y_w = r. \tag{4}$$

B.2 Dynamical model of the mechanical system

We next proceed to derive a dynamical model for the robot in movement. A scheme of all forces acting on the robot is shown in Figure 4.

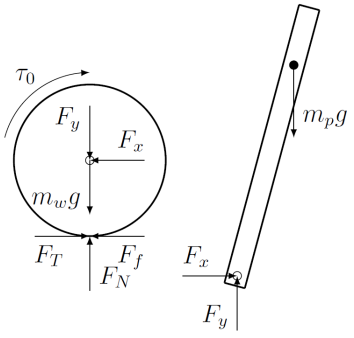

Figure 4: Forces acting on the robot.

The equations of motion of the wheel are:

$$m_w \ddot{x}_w = F_T - F_x - F_f, \tag{5a}$$
$$0 = F_N - F_y - m_w g, \tag{5b}$$
$$I_w \ddot{\varphi} = -rF_T + rF_f + \tau_0. \tag{5c}$$

The equations of motion of the pendulum are:

$$m_p \ddot{x}_p = F_x; \tag{6a}$$
$$m_p \ddot{y}_p = F_y - m_p g; \tag{6b}$$
$$I_p \ddot{\theta} = -F_x \ell \cos\theta + F_y \ell \sin\theta. \tag{6c}$$

Incorporating equations (6a) and (6b) into equation (6c), and substituting $\ddot{x}_p$ and $\ddot{y}_p$ (obtained by deriving twice (1a) and (1b)), the dynamic equation for the pendulum becomes:

$$I_p \ddot{\theta} + m_p \ell^2 \ddot{\theta} + m_p r \ell \ddot{\varphi} - m_p g \ell \sin\theta = 0. \tag{7}$$

Similarly, substituting equations (5a) and (5b) into equation (5c) and applying the time derivatives of equations (1a), (1b) and (2), it gives:

$$I_w \ddot{\varphi} + r^2 (m_w + m_p) \ddot{\varphi} + m_p r \ell \ddot{\theta} \cos\theta - m_p r \ell \dot{\theta}^2 \sin\theta = \tau_0. \tag{8}$$

These two models, equations (7) and (8), compose the **dynamical model of the mechanical part of the robot**. In words, they describe how a certain torque τ_0 (which we can control by actuating the motors attached to the wheels) modifies the state variables of the pendulum (namely, θ and its derivative) and those of the wheels (namely, ϕ and its derivatives).

The parameters of the mechanical model for the Pololu Balboa 32U4 are presented in Table 1.

Symbols	Description	Values	Units
m_w	Mass of the wheels	0.042	kg
m_p	Mass of the pendulum	0.316	kg
r	External radius of the wheels	0.040	m
r_i	Internal radius of the wheels	0.031	m
h	Height of the pendulum	0.109	m
d	Depth of the pendulum	0.022	m
ℓ	Distance between pendulum and wheel center of mass	0.023	m
I_p	Moment of inertia of the pendulum	$444.43 \cdot 10^{-6}$	kg m^2
I_w	Moment of inertia of the two wheels	$26.89 \cdot 10^{-6}$	kg m^2

Table 1: List of model parameters

B.3 Dynamical model of the electrical system

The wheels of the robots are actuated by two DC motors, whose scheme is illustrated in Fig. 5.

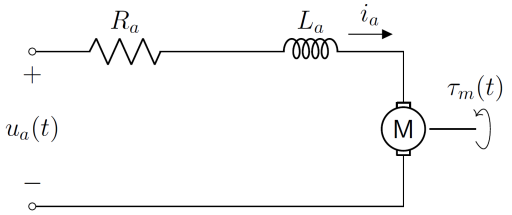

Figure 5: Electrical subsystem of the robot.

By applying Kirchhoff's laws to the DC motor's electrical circuit, we have:

$$u_a = R_a \cdot i_a + L_a \frac{di_a}{dt} + k_\phi \cdot \dot{\varphi}, \tag{9}$$

where k_ϕ is the back e.m.f. constant, and $\dot{\varphi}$ represents the motor's rotational speed. Since $L_a \approx 0$, it is reasonable to neglect the inductanctive term, which gives:

$$u_a = R_a \cdot i_a + k_\phi \cdot \dot{\varphi}. \tag{10}$$

The 'nominal" torque generated by a DC motor in the absence of frictions is proportional to the current:

$$\tau_m \approx C_{em} = k_\phi \cdot i_a. \tag{11}$$

However, in the presence of frictions, the nominal torque is reduced by a term proportional to viscous friction and by a term due to static friction:

$$\tau_m = C_{em} - K_\nu \cdot \dot{\varphi} - C_r \cdot \text{sgn}(\dot{\varphi}), \tag{12}$$

where K_ν represents the viscous friction coefficient and C_r denotes the static resistive torque. Intuitively, $C_r \cdot \text{sgn}(\dot{\varphi})$ models the minimum torque that needs to be generated by the motor before the wheels of the robot start moving, that is, to beat static frictions.

Finally, by substituting equations (11) and (12) into equation (10), and considering that the self-balancing system is operated by two motors (i.e., $\tau_0 = 2 \cdot \tau_m$), the input control voltage is given by:

$$u_a = \frac{R_a}{k_\phi}\left(\frac{\tau_0}{2} + K_\nu \cdot \dot{\varphi} + C_r \cdot \text{sgn}(\dot{\varphi})\right) + k_\phi \cdot \dot{\varphi}, \tag{13}$$

which can be expressed in compact form as:

$$u_a = a \cdot \tau_0 + b \cdot \dot{\varphi} + c \cdot \text{sgn}(\dot{\varphi}), \tag{14}$$

where we defined $a = \frac{R_a}{2k_\phi}$, $b = \frac{R_a K_\nu}{k_\phi} + k_\phi$ and $c = \frac{R_a C_r}{k_\phi}$.

In summary, we have found that to obtain a torque τ_0, we need to apply to the motor the voltage (14).

The parameters of the electircal model for the Pololu Balboa 32U4 are presented in Table 2.

Symbols	Description	Values	Units
$u_{a,max}$	Battery nominal voltage	7.2	V
R_a	Armature resistance	4	Ω
k_ϕ	Motor constant	0.132	Nm/A
K_ν	Viscous damping coefficient	$1.91 \cdot 10^{-3}$	Nm/(rad/s)
C_r	Resistance constant torque	$14.85 \cdot 10^{-3}$	Nm

Table 2: List of motor parameters

C Implementation of the model in Simulink

The first task to accomplish for this lab is to: **construct a MATLAB Simulink model for the "dynamical model of the mechanical system,"** corresponding to equations (7) and (8). Note that you will not need to model the electrical system for this task.

The resulting block model should have:

- inputs: the torque τ_0
- outputs: the variables φ, θ, $\dot{\varphi}$ and $\dot{\theta}$.

Questions

- Put an image of the block diagram (e.g., gains, integrators, sums, etc.) corresponding to the model of the Pololu Balboa 32U4 implemented in Simulink.
- Set the input corresponding to τ to 0, and test the nonlinear model with an initial angle $\theta = 10[^\circ]$ and $\theta = 45[^\circ]$. What do you obtain? Is it coherent with the expected behavior of the robot?

D Control of the balancing robot

In this section, we will design a control law for the robot and implement it in the real system.

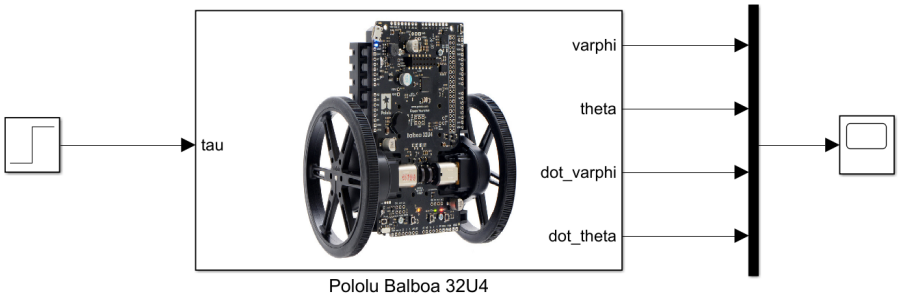

Figure 6: Block model of the robot in Simulink.

D.1 Linearization of the mechanical model

Our goal is to control the robot around its vertical position (i.e., $\theta = 0$) using linear control techniques. To do this, we first need to linearize the nonlinear mechanical model (eqs. (7)-(8)) around the equilibrium point $\theta^* = 0$.

Using the approximations $\cos\theta \approx 1$ and $\sin\theta \approx \theta$ valid for $\theta \approx 0$, the linearized version of (7)-(8) is:

$$E \begin{bmatrix} \ddot{\varphi}(t) \\ \ddot{\theta}(t) \end{bmatrix} + G\theta(t) = F\tau_0(t), \tag{15}$$

where:

$$E = \begin{bmatrix} I_w + r^2(m_w + m_p) & m_p r\ell \\ m_p r\ell & I_p + m_p\ell^2 \end{bmatrix}, \qquad G = \begin{bmatrix} 0 \\ -m_p g\ell \end{bmatrix}, \qquad F = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

In this system, E represents the inertia matrix, G the gravitational vector and F the control vector of the robot. Then, by defining the state vector:

$$x(t) = \begin{bmatrix} \varphi(t) \\ \theta(t) \\ \dot{\varphi}(t) \\ \dot{\theta}(t) \end{bmatrix},$$

we obtain the linear state space representation:

$$\dot{x} = Ax + B\tau_0 \tag{16}$$

where:

$$A = \left[\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 0 & \multirow{2}{*}{$-E^{-1}G$} & 0 & 0 \\ 0 & & 0 & 0 \end{array}\right], \qquad B = \left[\begin{array}{c} 0 \\ 0 \\ \hline E^{-1}F \end{array}\right],$$

D.2 State feedback control

In this section, we will use state feedback to balance the robot or, equivalently, to control the pendulum towards its vertical equilibrium. Consider the following control loop for the linearized model:

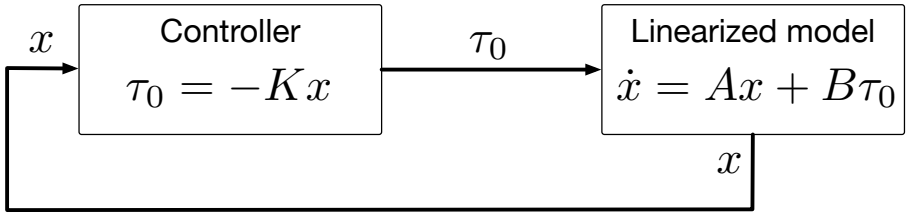

Figure 7: Closed-loop control system of the robot.

Following the diagram, the state feedback law to be applied is:

$$\tau_0(t) = -Kx = -K \cdot \begin{bmatrix} \varphi(t) \\ \theta(t) \\ \dot{\varphi}(t) \\ \dot{\theta}(t) \end{bmatrix}. \tag{17}$$

To choose the closed-loop poles, we will consider two pole placement methods:

Bessel pole locations: based on Bessel's method, the desired pole locations for a system with four state variables and a specified settling time t_s should be placed at:

$$\mathcal{L}_{Bessel} = \frac{1}{t_s} \cdot \{-4.016 + 5.072i, -4.016 - 5.072i, -5.528 + 1.655i, , -5.528 - 1.655i\}. \tag{18}$$

Integral of Time-weighted Absolute Error (ITAE) pole locations: based on the ITAE method, the desired pole locations for a system with four state variables and a specified settling time t_s should be placed at:

$$\mathcal{L}_{ITAE} = \frac{1}{t_s} \cdot \{-4.236 + 12.617i, -4.236 - 12.617i, -6.254 + 4.139i, -6.254 - 4.139i\}. \tag{19}$$

The second task to accomplish for this lab is to: **use MATLAB to compute the state feedback matrix K_{Bessel} such that the closed-loop poles are placed at $\mathcal{L}_{Bessel}$, and the state feedback matrix K_{ITAE} such that the closed-loop poles are placed at $\mathcal{L}_{ITAE}$.**

For the value of t_s, you are asked to consider two values: $t_s = 0.8$[s] and $t_s = 1.$#group[s] (e.g., group 1: $t_s = 1.01$[s], group 38: $t_s = 1.38$[s]).

To obtain K, you have to use the function `place(A,B,p)` in MATLAB, where p is the vector with the desired poles.

Put the values of K_{Bessel} and K_{ITAE} in your report.

D.3 Simulation of the controller in Simulink

The third task to accomplish for this lab is: **apply the two controllers you obtained above to the Simulink model you constructed.**

Questions

- Test the different gains K_x with an initial value $\theta = 10$[°] and $\theta = 90$[°]. Does the system remain at the equilibrium point $\theta^* = 0$?
- Compare the results obtained with Bessel and ITAE. Which one has a better performance?
- Based on the control law u_a, justify why the modeling of the electrical part in Simulink does not provide any information.
- Notice that you designed a controller for a linear model, but you are applying it now to a nonlinear model. Comment on this aspect.

D.4 Reference tracking (Bonus part)

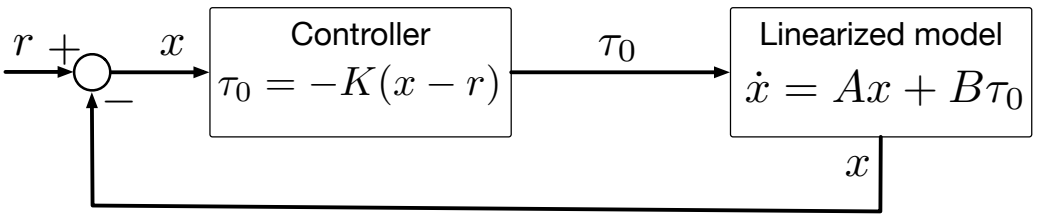

Figure 8: Reference tracking loop.

So far, you have designed a controller to balance the robot, but the robot remains stationary without advancing from its departure point. To have the robot advance from its initial position, consider the feedback loop in Fig. 8. Following the block diagram, and because making the robot advance corresponds to modifying $\varphi(t)$ only, the feedback law now becomes:

$$\tau_0(t) = -K_x \cdot \begin{bmatrix} \varphi(t) - r(t) \\ \theta(t) \\ \dot{\varphi}(t) \\ \dot{\theta}(t) \end{bmatrix}. \tag{20}$$

The task here is: **adjust your Simulink model to account for the additional reference signal.**

Your diagram should look similar to Fig. 9. For the reference of the trajectory use the block **Pulse generator** of Simulink with appropriate values (free choice).

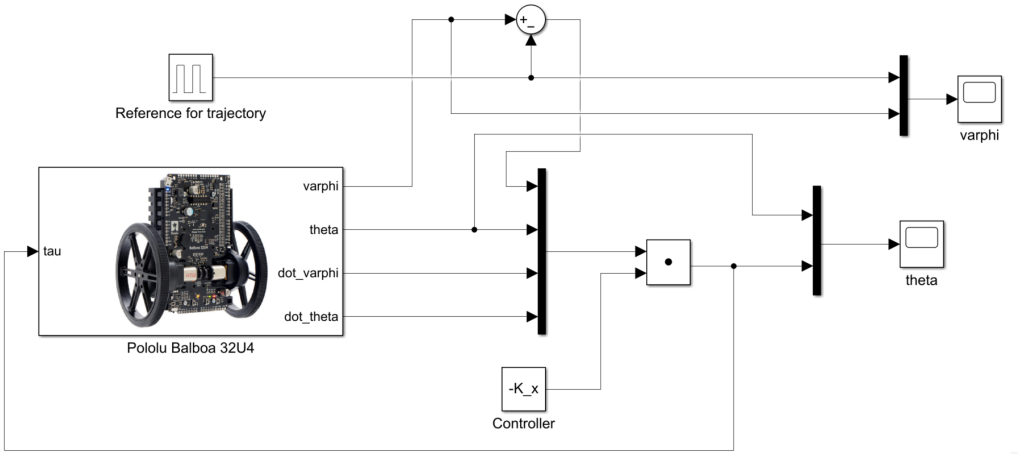

Figure 9: Block model of the robot in Simulink with the full state feedback control.

D.5 Preparation for the implementation on the real-world robot

Since for the physical robot we cannot actuate directly τ_0, but we can only decide what voltage is applied to the motors onboard, we need to translate torque inputs into voltage inputs. This can be done using (14):

$$u_a = a \cdot \tau_0 + b \cdot \dot{\varphi} + c \cdot \mathrm{sgn}(\dot{\varphi}).$$

For this reason, when you will implement your controller on the real robot, your controller will be the cascade of the state feedback controller you designed above and the torque-voltage conversion map. This is illustrated in Figure 10.

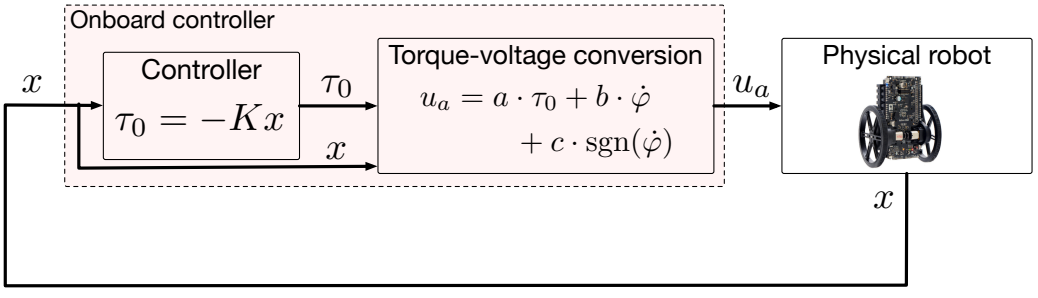

Figure 10: Block diagram of the real robot with torque-voltage conversion.

E Arduino setup

It is now time to implement the controllers you derived on the real robot.

Important

Those robots are **expensive and fragile**. We ask you to be gentle with them, and to avoid to make them fall or to **damage them in any way**. The robots must be tested exclusively in the safe area depicted in Figure 2.

To proceed, we ask you to use your personal computer for this laboratory. It must have a USB port with a type A connector (to plug the cable to link the robot and your computer), or a converter that has such a port. If for any reason you cannot use a personal computer, please contact the teaching assistants[2].

First, you have to download the **Arduino IDE**. Go to `https://www.arduino.cc/en/software`, download the latest Arduino IDE for your personal setup and install it on your computer. Open the IDE on your computer.

Then, follow these steps.

1. Go to **Boards manager**, and install **Arduino AVR Boards** (by Arduino).
2. Go to **Library manager**, and install **LSM6** (by Pololu), and **Balboa32U4** (also by Pololu).
3. Open **File** > **Examples** > **Balboa32U4** > **Balancer**.
4. Save **Balancer** on your computer with `Cmd/Ctrl` + `S`.

Now, you are ready to upload the algorithm on the robot. To do that, follow these steps.

1. Connect the robot to your computer with the given USB cable. You should see **Arduino Leonardo** appear in the list on the upper left corner of the IDE, such as shown in Figure 11. Select it. If it didn't appear, manually select the appropriate COM port and specify **Arduino Leonardo** as a board.
2. Click on the **Upload** button on the left of the list. It is the arrow in Figure 11.
3. Wait for the message "Done uploading" in the IDE, and make sure that the light is green on the board of the robot.

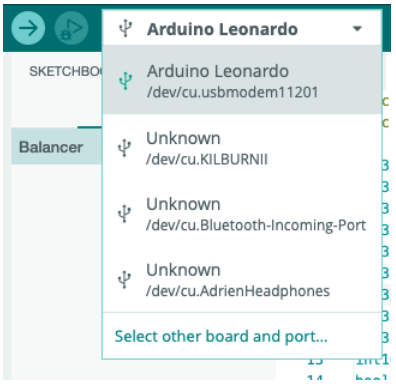

Figure 11: The Arduino Leonardo board appears next to other boards/ports such as Bluetooth and USB ports.

You are now ready to **test the robot**! To make the robot balancing, follow these steps.

1. If connected, disconnect the USB cable.
2. Place the robot inside its rolling area, in the same position as in Figure 12.
3. Make sure that the toggle switch is at the left position, such as Figure 13.
4. Click on the button just above the toggle switch, and wait for the green light.
5. Gently put the robot on balance position such as illustrated in Figure 14, and see the magic happening!

F Experiments description

This section contains the description of the experiments that you have to perform. First, we will ask you to **draw a parallel** between the implemented controller and a classical control strategy seen in class. Second,

[2] `renato.vizueteharo@uclouvain.be` and `francois.wielant@uclouvain.be`

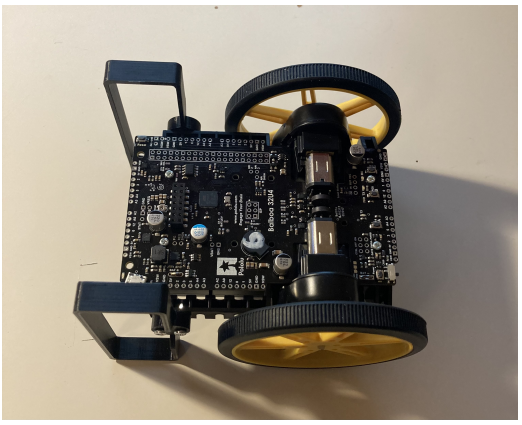

Figure 12: Initial position of the robot.

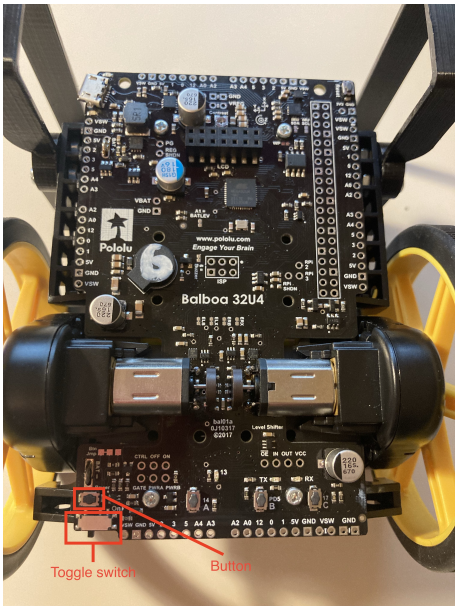

Figure 13: Board of the robot, with the concerned buttons highlighted.

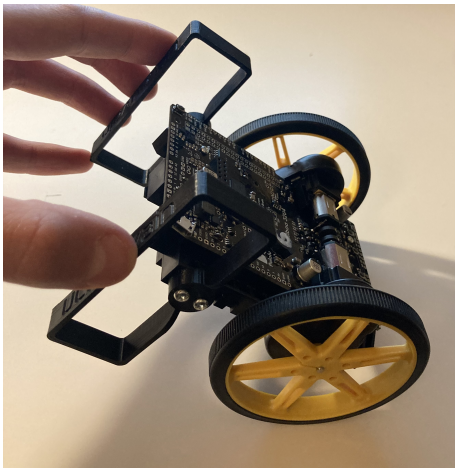

Figure 14: The robot is put on balance position.

we will ask you to **implement** the full state feedback controller designed in previous sections and modify the desired trajectory.

F.1 Identification of classical strategies

First, go to `Balance.cpp` and have a look at the `balanceUpdate` function at line 184. This function is called from `Balancer.ino` in the `loop` function that constantly runs when the robot is powered on. In particular, have a closer look at the `balance` function at line 59.

To help you understand the code you will find below a list of the main variables to know and their signification for the real system.

- `angle` [mdeg] - The angle of the body relative to vertical;
- `angleRate` [deg/s] - The rate of rotation, or rational speed of the body;
- `distanceLeft/distanceRight` [**some unit**] - Encoder-based distance from original position of the left-/right wheel;
- `speedLeft/speedRight` [**some unit**] - Encoder-based speed of left/right wheel;

Questions

- Explain in a few lines how the implemented controller works.
- Draw a parallel between those functions and the full state feedback controller **seen in class**. If the latter can be divided into different parts, make the link between every part and the implemented controller.

Important

Please provide answers using **your own words**. Copy/pasting from any other source will be severely penalized.

F.2 Implementation of state feedback control

For this section, you have to open the project `State Feedback` (available on Moodle). In the function `balance` of the file `Balance.cpp` of this project you have to introduce the gains obtained with Bessel and ITAE for $t_s = 1.2[s]$ (remember that you must use the matrix B' instead of B). Next, you have to upload the code to the robot and test the behavior. Try to gently move the robot from the equilibrium position as a disturbance to verify the performance of the controller.

F.3 Implementation of reference tracking control

The objective now is to adjust your controller to implement the reference tracking controller (20). The goal is to make the robot move forward 50[cm] in 2[s]. Then, it must stay in that position during 3[s] and return to the original position in 2[s], where it must stay again during 3[s]. Finally, the robot must follow the same trajectory again (i.e., a repeating sequence). To do this, you have to modify the output ua in the function `balance` and the parameters of the function `refTrack`.

Questions

- Why does the gain $K3$ have an additional value?
- Write the code that you modified for the implementation of the trajectory tracking.
- Compare again the results obtained with Bessel and ITAE. Which one has a better performance? Is it coherent with the simulation?

G Deliverable

You are asked to write a **one-page report** of the experiment (including student names and all the necessary information, reports longer than one page will not be accepted), in which you **provide an answer to all the questions in the green boxes above**.

The summary should be **concise but complete**, and should **reflect your understanding** of the laboratory material **as much as possible**. It must contain **at least** the following items.

- The NOMA and names of all the members of the group;

- An answer to every question in the green boxes;

You have to send **one copy of the report per group** on the corresponding Moodle activity. **Each report is due on 'day'+1 at 23:59, where 'day' is the day you completed the lab experience**. No report submitted after this deadline will be considered for correction.

Contact

For any practical or theoretical question, please contact:

Renato Vizuete (Office: A.-116): renato.vizueteharo@uclouvain.be

François Wielant (Office: A.-122): francois.wielant@uclouvain.be