Control Systems Laboratory guide

for the 1st quarter of 2023/2024

Rui Coelho, Alexandra Moutinho, André Serra, Duarte Valério

October 18, 2023

Contents

Introduction

Theoria sine praxi, rota sine axi.
Praxis sine theoria, cæcus in via.

Medieval proverb

Objectives

In accordance with the proverb above, laboratory classes have two purposes:

- to impress upon students that subjects studied in this course are no mathematical abstraction, but rather correspond to real things in the real world;
- to help students gain experience and insight in the not-always-straightforward task of studying a plant and designing its control system.

Laboratory Assignments

Laboratory classes take place in the Control, Automation and Robotics Laboratory, Mechanical Engineering III building, basement floor 1. Laboratory work is carried out in groups of three students, at most. There are 5 laboratory assignments for the students to complete. These assignments are described in the different chapters of this guide and follow the schedule below:

Lab Assignment	Guide Chapter	Lab Class Week
Lab 1	Chapter 1	Week 3 (25/09/2023-01/10/2023)
Lab 2	Chapter 2	Week 4 (02/10/2023-06/10/2023)
Lab 3	Chapter 3	Week 5 (09/10/2023-13/10/2023)
Lab 4	Chapter 4	Week 6 (16/10/2023-20/10/2023)
Lab 5	Chapter 5	Week 7 (23/10/2023-27/10/2023)

The students must prepare for each lab beforehand, by reading the corresponding lab guide chapter and performing the necessary tasks there described. Failing to do this previous preparation will compromise the lab class (and hence the grade).

Laboratory work will be graded according to the experimental work and by means of an online quiz that each student will have to answer at the end of the laboratory session.

Marks

Each student will receive a mark in a 0–5 scale for every laboratory class attended. The final mark of the laboratory component of the Control Systems course, in a 0–20 scale, is the sum of four best laboratory grades (out of the five partial marks possible). To pass the course, this sum must be 9.5 or higher. It accounts for 30% of the final mark of the course.

The grade of a laboratory is divided into:

- 0–2 values: determined by the lab professor depending on your group performance;
- 0–3 values: defined by your individual response to an online quiz in the end of the lab.

Missing a class means a zero grade for that class.

Students enrolled in previous academic years

Marks obtained in previous academic years will not be considered.

Students entitled to special evaluation period and with Working-Student status

Students entitled to a special evaluation period and with Working-Student status are subject to the rules above. Students entitled to a special evaluation period may attend classes during the first quarter, or, if unable to, they may have laboratory classes during the special evaluation period, appointed by agreement, during the second half of July 2024.

Chapter 1

Modelling the plant

The objective of this Chapter is to present and model the "Rasteirinho", a mobile robot which will be the object of the laboratory work in the Control Systems course.

1.1 Preparing the class beforehand

Before class, you should:

- read carefully the description of the Rasteirinho below;
- download the MATLAB/Simulink files needed for the lab, as listed in Section 1.1.3.

1.1.1 The Rasteirinho

While airplanes and ships have, for a long time now, the benefit of automatic pilots, cars are achieving a comparable degree of autonomy only these days. And yet the dream of a car that drives itself to its destination, with minimum human intervention, safely, comfortably and economically, can be already found in both the scientific and the science fiction literature of the mid-20th century (see e.g. Isaac Asimov's 1953 short story *Sally*).

Of course, self-driving cars, buses, lorries, water vessels, or aircraft would be utterly impossible without complex, high-performing control systems. While it is clearly unfeasible to replicate cutting-edge research in this field during a one-quarter introductory course on control systems, the basic ideas thereof can be explored in the laboratory, using a mobile robot controlled by a computer running Matlab.

The mobile robot we will be working with is called "Rasteirinho", and is shown in Figure 1.1 together with its major parts. It is composed of two motorized wheels (and one free-wheel).

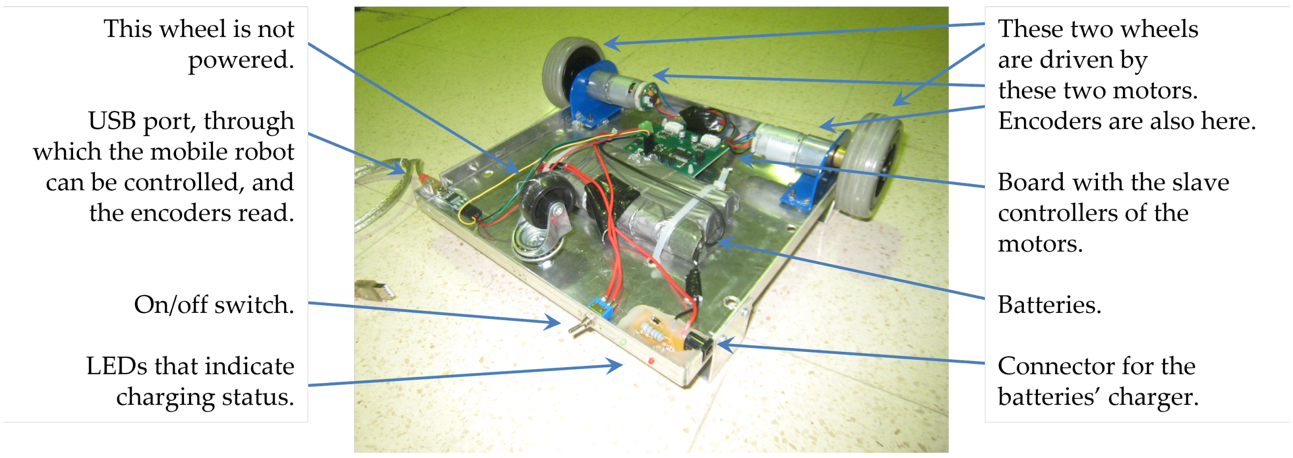

This wheel is not powered.
USB port, through which the mobile robot can be controlled, and the encoders read.
On/off switch.
LEDs that indicate charging status.
These two wheels are driven by these two motors. Encoders are also here.
Board with the slave controllers of the motors.
Batteries.
Connector for the batteries' charger.

Figure 1.1: The Rasteirinho, seen from below. It can be seen from above in this guide's frontispiece. Front (powered) wheels radius: $r = 0.0365$ m; Distance between front wheels: $L = 0.295$m.

1.1.2 How the Rasteirinho is controlled

The Rasteirinho has a MD25 motor driver which controls the angular velocity of the two wheels, according to the command signal sent from the computer, and reads the encoder values from each wheel.

Through a USB (Serial) port, we send two digital signals, β_r and β_l, to command the velocity of the right and left motors of the Rasteirinho, respectively. These signals are represented by 8-bit, signed, integer values, ranging from -128 to 127. Value 0 means that the motor is stopped. Value 127 means that the motor runs at maximum speed in one direction. Value -128 means that the motor runs at maximum speed in the other direction. The angular velocity of the wheel, ω_{wheel}, will be proportional to the command signal sent, β_{wheel}:

$$\omega_{wheel} = K\beta_{wheel}, \tag{1.1}$$

where $K = 0.1581$rad/s.

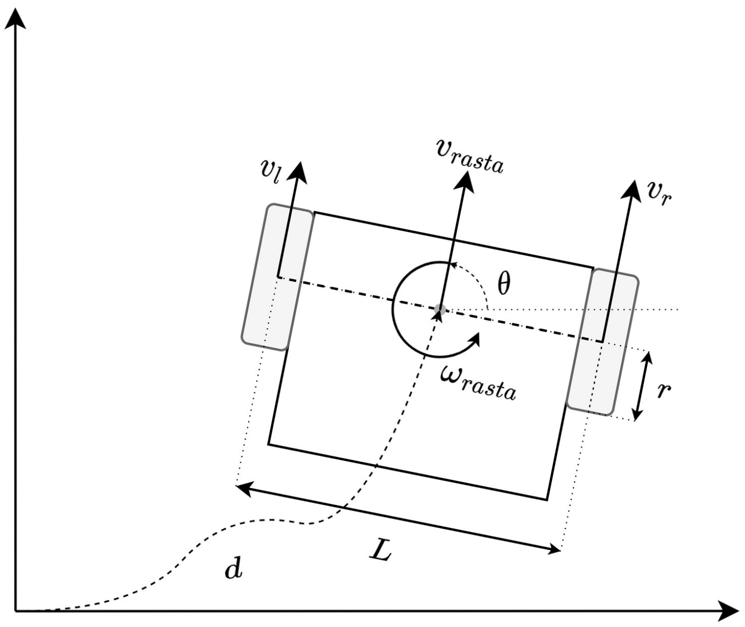


Figure 1.2: Depiction of the Rasterinho movement.

Taking the wheel radius $r = 0.0365$ m, the linear velocity of a wheel is

$$v_{wheel} = \omega_{wheel} r = Kr\beta_{wheel} \tag{1.2}$$

The linear velocity of the Rasteirinho is given by the average linear velocity of the two wheels:

$$v_{rasta} = \frac{v_l + v_r}{2} = Kr\frac{\beta_l + \beta_r}{2} = \alpha_d\bar{\beta}, \tag{1.3}$$

where $\alpha_d = Kr$ and $\bar{\beta} = \frac{\beta_l+\beta_r}{2}$.

Considering the distance between wheels, $L = 0.295$m, the angular velocity of the Rasteirinho, ω_{rasta} is

$$\omega_{rasta} = \frac{v_r - v_l}{L} = \frac{2Kr}{L}\frac{\beta_r - \beta_l}{2} = \alpha_\theta\Delta\beta, \tag{1.4}$$

where $\alpha_\theta = \frac{2Kr}{L}$ and $\Delta\beta = \frac{\beta_r-\beta_l}{2}$.

The displacement, d, of the Rasteirinho along a path can be retrieved by integrating its linear velocity:

$$\begin{aligned} \dot{d}(t) &= v_{rasta}(t) = \alpha_d \bar{\beta}(t) \\ \Longrightarrow sD(s) &= \alpha_d \bar{\beta}(s) \Longrightarrow D(s) = \frac{\alpha_d}{s}\bar{\beta}(s) \end{aligned} \tag{1.5}$$

Similarly, the angular displacement, θ, is obtained by integrating the angular velocity of the Rasteirinho:

$$\begin{aligned} \dot{\theta}(t) &= \omega_{rasta}(t) = \alpha_\theta \Delta\beta(t) \\ \Longrightarrow s\Theta(s) &= \alpha_\theta \Delta\beta(s) \Longrightarrow \Theta(s) = \frac{\alpha_\theta}{s}\Delta\beta(s) \end{aligned} \tag{1.6}$$

The USB port is also used to read the values of the encoders connected to each wheel. The 32-bit encoder reads values between $[-2^{31}, 2^{31}-1]$. The encoder reads 360 pulses per wheel turn (1 pulse=$1° = \frac{\pi}{180}$rad). When the program starts, encoders are reset to 0. Turning the wheels forward results in an increment of the encoders, whereas turning them backward results in a decrement of the encoders. The linear distance travelled by a wheel, d_{wheel}, is proportional to its encoder reading, ρ_{wheel}:

$$d_{wheel} = r\frac{\pi}{180}\rho_{wheel} \tag{1.7}$$

Odometry takes the encoder readings, ρ_l and ρ_r, from the left and right wheels, respectively, to measure the linear (d_{odom}) and angular (θ_{odom}) displacement of the Rasteirinho:

$$d_{odom}(t) = \frac{d_l(t) + d_r(t)}{2} = r\frac{\pi}{180}\frac{\rho_l(t) + \rho_r(t)}{2} \tag{1.8}$$

$$\theta_{odom}(t) = \int_0^t \omega_{rasta}(\tau)\, d\tau = \frac{1}{L}\int_0^t v_r(\tau) - v_l(\tau)\, d\tau = \frac{1}{L}(d_r(t) - d_l(t)) = \frac{r}{L}\frac{\pi}{180}(\rho_r(t) - \rho_l(t)) \tag{1.9}$$

The Rasteirinho is a multiple-input, multiple-output (MIMO) plant, with two inputs and two outputs. Fortunately, we can treat it as two decoupled single-input, single-output (SISO) plants, as described in Eq. (1.5) and (1.6). From the control actions $\bar{\beta}$ and $\Delta\beta$, the command signals sent to the left and right wheels, β_l and β_r, respectively, are computed as follows:

$$\bar{\beta} = \frac{\beta_l + \beta_r}{2} \tag{1.10}$$

$$\Delta\beta = \frac{\beta_r - \beta_l}{2} \tag{1.11}$$

$$\beta_l = \bar{\beta} - \Delta\beta \tag{1.12}$$

$$\beta_r = \bar{\beta} + \Delta\beta \tag{1.13}$$

We need to implement controllers for both the displacement of the Rasteirinho and its attitude. This is how we will do it:

- We will first validate that the models for the displacement and attitude follow the linear laws described in Eq. (1.5) and (1.6), comparing simulation results to experimental data.
- In Chapter 2, we then set $\Delta\beta = 0$, meaning that the Rasteirinho will only move forward in a (more or less) straight line, without turning right or left. As the model we will find in this Chapter is fairly simple, we will control this longitudinal movement using open-loop control.
- In Chapter 3, and since open-loop control has significant limitations, we will then control the longitudinal movement using closed-loop proportional control with the readings from the encoders.
- In Chapter 4, we will also control the attitude in closed-loop so that the Rasteirinho follows different references in both displacement and attitude. These references can be circles, squares, rectangles or any specific path set for it to follow.
- In Chapter 5, we will again control the Rasteirinho longitudinal and lateral motions in closed-loop, but now using the optical tracking system in the Robotics Arena. Instead of using the encoders, the position and orientation of the Rasteirinho will be measured by this external system.

1.1.3 Software

Download all the necessary software from the course webpage to the laptop you will be using to control the Rasteirinho. Make sure to have the Matlab version indicated installed. If you are not very familiar with Simulink and Matlab, there are several resources online where you can learn about the basics, such as Simulink Onramp and Matlab Onramp.

1.2 Laboratory preparation

This section helps you create a Simulink model of the Rasteirinho, that will allow you to simulate the vehicle behavior and later design and test the controllers that will be applied to the real Rasteirinho.

In order to model the displacement and attitude of the Rasteirinho, we can set two different transfer functions in order to mimic its behaviour. To determine them the following set of questions serves as a guide in order to obtain those functions. The parameters already provided are for a specific Rasterinho, so there may be small differences between different cars.

1. Determine the values of α_d and α_θ.
2. Find the ranges of values that $\bar{\beta}$ and $\Delta\beta$ can assume.
3. Find the maximum speeds of the Rasteirinho. Notice that these are a consequence of the inputs (also known as actuation) limits or saturations.
4. Find transfer function $G_d(s)$, relating the displacement of the Rasteirinho measured in meters (output) with $\bar{\beta}$ (input / actuation).
5. Find transfer function $G_\theta(s)$, relating the attitude of the Rasteirinho measured in rad (output) with $\Delta\beta$ (input / actuation).
6. Create a new Simulink file (name it e.g. `rastamodel.slx`), where you will implement your Rasteirinho simulator. Implement the transfer functions $G_d(s)$ and $G_\theta(s)$ in this Simulink file. You can use the `Transfer Fcn` block.
7. Implement Eqs. (1.10) and (1.11) in Simulink, to convert from control variables β_l and β_r to $\bar{\beta}$ and $\Delta\beta$ (use, for example, the `Interpreted MATLAB Fcn` block).
8. Implement a step input to each of the control variables, $\bar{\beta}$ and $\Delta\beta$ using the `Step` block.
9. Remember to impose the computed limitations to the control variables input using the `Saturation` block.
10. Implement `Scope` blocks to see the inputs / outputs' graphics
11. You will find the blocks you need in Simulink's libraries `Continuous`, `Sources`, `User-Defined Functions`, `Commonly Used Blocks`, `Discontinuities` and `Sinks`, or typing their name in your Simulink model.
12. Provide equal nonzero steps to β_l and β_r (see what would be a reasonable value) and check if the respective outputs are in agreement with your expectations (the car should do a straight line at a constant speed, i.e., θ should be zero).
13. Provide different nonzero steps to β_l and β_r (see what would be a reasonable value) and check if the respective outputs are in agreement with your expectations (the car should not do a straight line, i.e., θ should not be zero).

1.3 In the laboratory

This section regards the work you will do in the lab, where you will adapt the provided Simulink `rasta.slx` so you can work with the Rasteirinho directly.

Notice that the purpose of this lab is that you have a good model/simulator of your car, so the results you obtain with your model should be similar to the ones you obtain with the real vehicle.

To connect to the Rasteirinho, follow these steps:

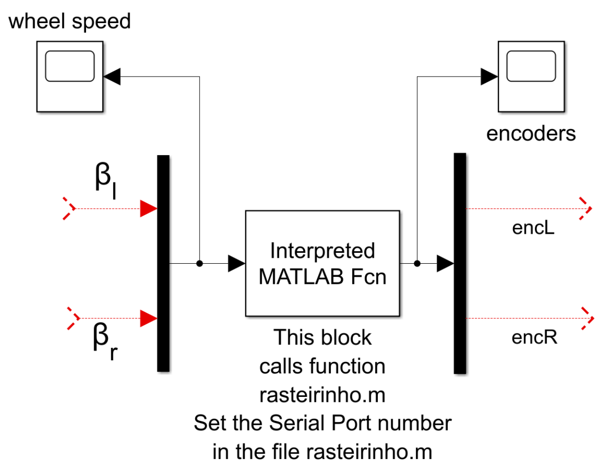

wheel speed
encoders
β_l
β_r
Interpreted MATLAB Fcn
This block calls function rasteirinho.m
Set the Serial Port number in the file rasteirinho.m
encL
encR

Figure 1.3: Simulink model to command the Rasteirinho (the real one) and read its encoder values.

1. Connect the USB cable to your computer. Verify which COM port is in use in `Control Panel > Device Manager`.
2. Turn the Rasteirinho on (verify that a red LED is now lit in the integrated circuit below the vehicle).
3. Open Matlab and set the current working folder to the directory where you keep the files for this class.
4. Open the Simulink file `rasta.slx` (see Figure 1.3). This Simulink file makes use of the code in `rasteirinho.m`, so open that file too and set the correct COM port as identified in code line 3.

Once you finished the setup, do the following tasks:

1. Implement Eqs.(1.8) and (1.9) to compute the displacement and orientation of the Rasteirinho from the encoder readings. Notice that this is an estimation of the real values, and may differ from these.
2. Implement Eqs.(1.12) and (1.13) to compute the command signals to the left and right wheels, β_l and β_r, from the control actions, $\bar{\beta}$ and $\Delta\beta$.
3. Similarly to what was done in simulation, apply a constant input to both wheels, $\beta_l = \beta_r$ ($\bar{\beta}$ =constant, $\Delta\beta = 0$), for a certain amount of time, and record the odometry-measured displacement of the robot, $d(t)$. Measure the real displacement using a measuring tape. Apply that same input to your simulated model and compare the experimental and simulated results.
4. Apply a symmetric input to both wheels, $\beta_l = -\beta_r$ ($\bar{\beta} = 0$, $\Delta\beta$ =constant), for a certain amount of time, and read the odometry-measured orientation of the robot, $\theta(t)$. Apply that same input to your simulated model and compare the experimental and simulated results.
5. Looking at the plots of displacement and orientation vs time of your real and simulated vehicles, can you improve the model parameters α_d and α_θ for your Rasteirinho?

Chapter 2

Open loop control of the longitudinal movement

To recall, in Lab 1 you developed a model of the Rasteirinho, which by the respective transfer functions, models the longitudinal and lateral motions of the robot. That model, represented in a simplistic way in the upper block diagram in Figure 2.1, was implemented in Simulink so that you can simulate the Rasteirinho at home when you do not have the Rasteirinho with you. You also prepared a Simulink file that communicates with the real Rasteirinho, which is also represented in a simplistic way in the lower block diagram in Figure 2.1, and that you use in the lab. Notice that both block diagrams (the one with the model and the one that uses the real Rasteirinho) have the same inputs (commands for $\bar{\beta}(t)$ and $\Delta\beta(t)$) and the same outputs ($d(t)$ and $\theta(t)$).

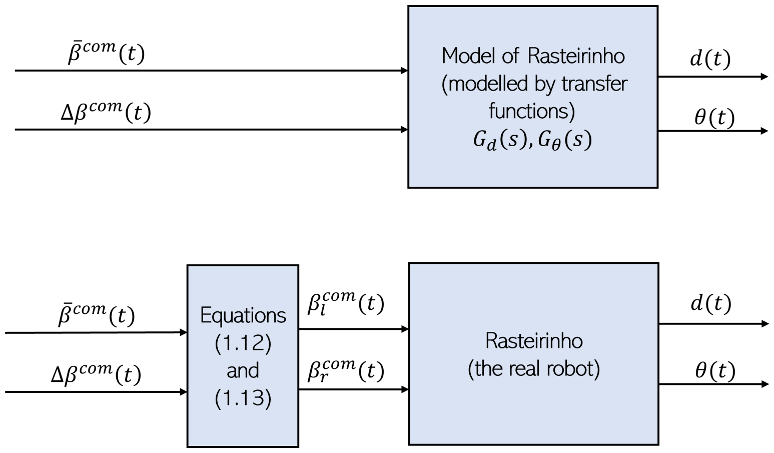


Figure 2.1: Simplistic representations of the simulated and real Rasteirinho systems (Lab 1).

The objective of Laboratory 2 is to show the advantages of open-loop control and the conditions in which it can be applied — as well as the disadvantages it has. Towards that end, you will design an open-loop controller for the longitudinal motion of the Rasteirinho (we will consider the $\Delta\beta(t)$ input zero), something like what is depicted simplistically for the Rasteirinho model and the real vehicle in Figure 2.2.

2.1 Laboratory preparation

Before the laboratory, you will work with the Simulink model of the Rasteirinho towards an open-loop controlled system able to follow a displacement reference $d_{ref}(t)$, following these steps:

1. Download the files corresponding to Lab 2 from the course webpage. We provide you the Simukink models you probably obtained in Lab 1 just to guarantee you all start with the same baseline.
2. Run the function `reference_builder.m` (type `ref = reference_builder` on the Matlab command window) to get the displacement reference, $d_{ref}(t)$, you will have to follow in the laboratory. Notice this will

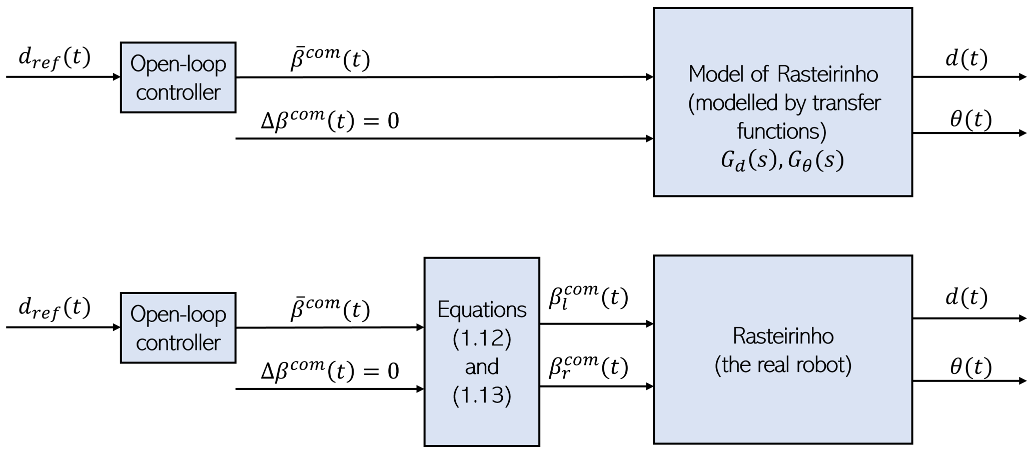

$d_{ref}(t)$
Open-loop controller
$\bar{\beta}^{com}(t)$
$\Delta\beta^{com}(t) = 0$
Model of Rasteirinho (modelled by transfer functions) $G_d(s), G_\theta(s)$
$d(t)$
$\theta(t)$
$d_{ref}(t)$
Open-loop controller
$\bar{\beta}^{com}(t)$
$\Delta\beta^{com}(t) = 0$
Equations (1.12) and (1.13)
$\beta_l^{com}(t)$
$\beta_r^{com}(t)$
Rasteirinho (the real robot)
$d(t)$
$\theta(t)$

Figure 2.2: Simplistic representations of the simulated and real Rasteirinho open-loop control systems (Lab 2).

provide the $d_{ref}(t)$ input to your open-loop controller.

3. Use the `From Workspace` block to read the variable `ref`, which contains the reference displacement $d_{ref}(t)$, from Matlab's Workspace into Simulink.

4. Devise an open-loop controller for the displacement of Rasteirinho, so as to follow (as much as possible) the reference $d_{ref}(t)$ given by `ref`. You may start by simply considering a gain, but if you remember from classes, there is probably a better open-loop controller so that the output better follows the input. Remember also that any controller needs to be proper. Add the designed open-loop controller to the Simulink model of the Rasteirinho, remembering it will receive $d_{ref}(t)$ and it will output a command $\bar{\beta}^{com}(t)$.

5. You will also need a command $\Delta\beta^{com}(t)$. In this case, since we are only interested in the longitudinal motion (no change in attitude), you will consider $\Delta\beta^{com}(t) = 0$. You can either use a step input (with zero final step value) or a constant input equal to zero.

6. Remember that the command signals sent to the left and right wheels may saturate. In this scenario, where $\Delta\beta = 0$, $\bar{\beta} = \beta_l = \beta_r$, so if you saturate $\bar{\beta}$ within its maximum range [-128, 127] in your simulated model, it is equivalent to saturating β_l and β_r. More generally, in case both control actions $\bar{\beta}$ and $\Delta\beta$ are nonnull, it is not enough to saturate each of them individually to guarantee that β_l, β_r will not saturate (e.g., what happens if $\bar{\beta} = 126$ and $\Delta\beta = 10$?). In that case, you would have to convert $(\bar{\beta}, \Delta\beta)$ to (β_l, β_r), saturate (β_l, β_r), and convert back to $(\bar{\beta}, \Delta\beta)$ (see Figure 2.3), so that your simulated model accurately represents what happens in the Rasteirinho.

7. Check the open-loop controlled system performance using the Simulink simulation, plotting:

   (a) The reference displacement $d_{ref}(t)$ vs the output $d(t)$

   (b) The reference attitude $\theta_{ref}(t)$ vs the output $\theta(t)$

   (c) The control action commands, $\bar{\beta}^{com}(t)$ and $\Delta\beta^{com}(t)$ vs the respective saturated inputs, $\bar{\beta}(t)$ and $\Delta\beta(t)$.

   Is the Rasteirinho model able to follow the reference? If not, why?

8. Add a disturbance (say, a low amplitude sinusoidal) to the output (i.e. displacement $d(t)$) of the simulation. Explain why the disturbance is not rejected.

9. Add a disturbance to the control action (i.e. $\bar{\beta}^{com}(t)$) of the simulation. Explain why the disturbance is not rejected.

10. Copy the appropriate parts you added to your Rasteirinho model in this section (except the disturbances, or consider then null) to the Simulink file that controls the real Rasteirinho, to test it later in the laboratory.

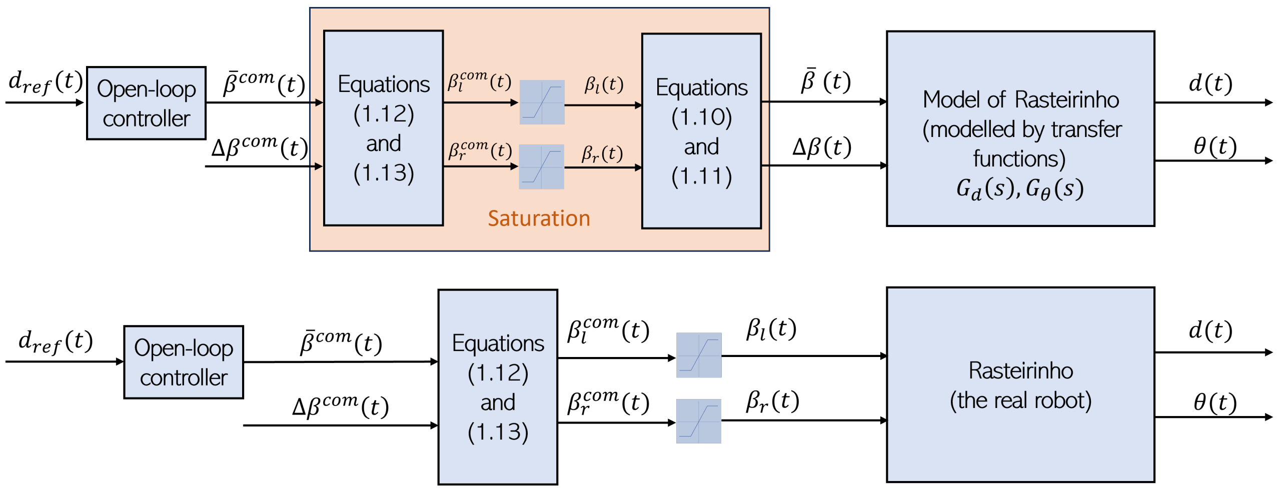


Figure 2.3: Emulating Rasterinho's input saturation in the simulated model.

2.2 In the laboratory

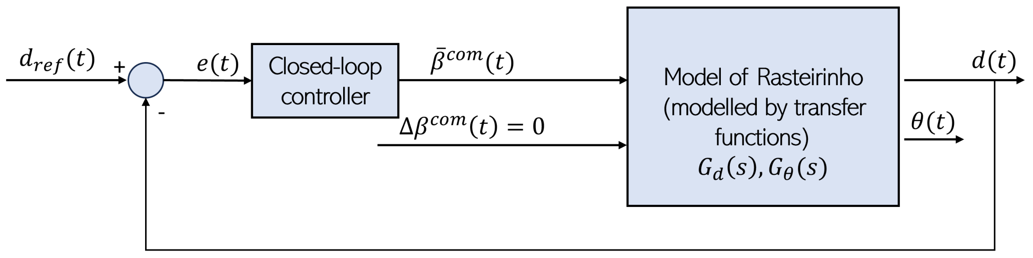
In the lab you will check if your open-loop controller applied to the real Rasteirinho will performed as simulated at home with the Rasteirinho model. Use the Simulink file that connects to the real robot and that you prepared in the last actions of the preparation work. Follow these steps at the lab:

1. Test your open-loop controller for the displacement control of the real Rasteirinho.
2. Plot, as functions of time, the distance actually travelled (as measured by the encoders), and the reference that you are trying to follow.
3. Show the instructor the Rasteirinho working with your controller and the results you got.

Chapter 3

Proportional control of the forward movement

In our previous session, Lab 2, we observed the inherent limitations of open-loop controllers. In order to refine the system's response to prescribed references and improve tracking performance, our objective in this session is to design a closed-loop control system. Specifically, we will introduce a Proportional (P) controller, where the output (displacement) is continually compared with the desired input (reference). The control action will be determined based on this error, ensuring precise error-driven adjustments for system actuation.



$d_{ref}(t)$
+
-
$e(t)$
Closed-loop controller
$\bar{\beta}^{com}(t)$
$\Delta\beta^{com}(t) = 0$
Model of Rasteirinho (modelled by transfer functions) $G_d(s), G_\theta(s)$
$d(t)$
$\theta(t)$

Figure 3.1: Simplistic representation of the simulated closed-loop control system (Lab 3).

The objectives of this Lab are:

- to show the advantages and disadvantages of closed-loop control, when compared with open-loop control;
- to show how the root locus can be used to design a proportional controller;
- to show how actuator saturation limits the performance of control systems.

3.1 Laboratory preparation

Before class, you should:

1. Download the files corresponding to Lab 3 from the course webpage. We provide you the Simukink models you probably obtained in Lab 1 just to guarantee you all start with the same baseline.
2. Plot the Bode and Nyquist diagrams, as well as the root locus of both $G_d(s)$ and $G_\theta(s)$. You can use Matlab commands `bode`, `nyquist` and `rlocus` for this.
3. Implement in simulation a closed-loop controller, consisting of plant $G_d(s)$ and a proportional controller K_d, with labels identifying where $\bar{\beta}$ and d are;
4. Analyse the root-locus of $G_d(s)$ and find a proportional controller, K_d, for the displacement of the Rasteirinho that satisfies (as much as possible) a settling time (2%) = 3.5s. Try several possibilities and check the performance using the Simulink simulation, using as references, $d_{ref}(t)$:

   (a) Step (with different amplitudes, e.g. 0.5m, 1m, 2m)
   (b) Ramp
   (c) Parabola
   (d) the reference you got from file `reference_builder.m`

5. Check the closed-loop controlled system performance using the Simulink simulation, plotting:
   (a) The reference displacement $d_{ref}(t)$ vs the output $d(t)$
   (b) The tracking error displacement $e(t) = d_{ref}(t) - d(t)$
   (c) The control action command, $\bar{\beta}^{com}(t)$ vs the saturated input, $\bar{\beta}(t)$.

   Is the Rasteirinho model able to follow the reference? If not, why?

   Add the disturbances you used in Lab 2 with open-loop control to your closed-loop simulation. Check if the disturbance is rejected, and explain why.

6. Copy the appropriate parts you added to your Rasteirinho model in this section (except the disturbances, or consider then null) to the Simulink file that controls the real Rasteirinho, to test the closed-loop controller in the laboratory.

3.2 In the laboratory

You must come to the laboratory knowing which proportional controller you will use.

1. Test your closed-loop controller for the displacement movement of the Rasteirinho using `rasta.slx`. Use the reference provided by file `reference_builder.m`.

2. Plot, as functions of time, the distance actually travelled (as measured by the encoders), the reference that you are trying to follow, and (in a separate plot) the control action.

3. Show the instructor the Rasteirinho working with your controller and the results you got.

Chapter 4

Controlling the attitude of a mobile robot

For the Rasteirinho to navigate freely around the world, it must be able to follow complex trajectories, which may include straight lines, curved paths, and more. In Lab 3, you have designed a closed-loop controller for the displacement of the Rasteirinho. The objective of this Lab 4 is to control the attitude of the Rasteirinho using a controller from the PID-family, and then control both attitude and displacement simultaneously (Figure 4.1).

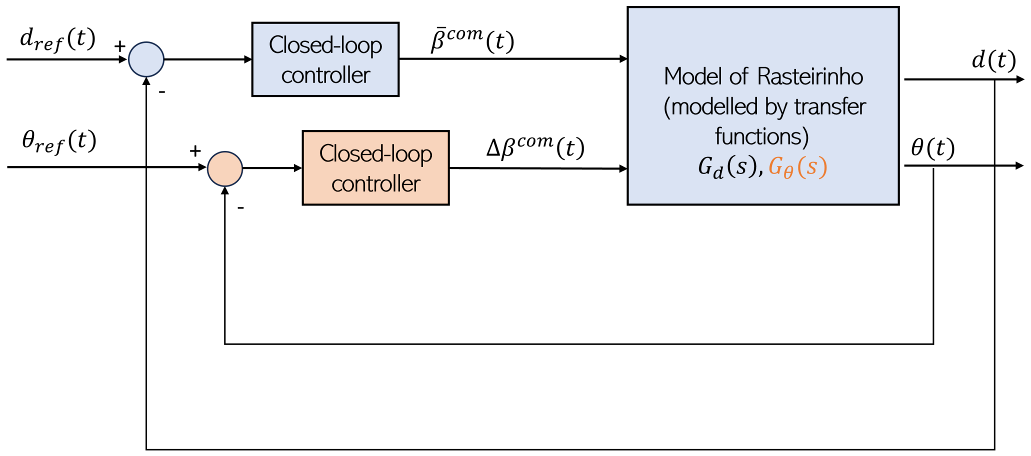


Figure 4.1: Simplistic representation of the simulated closed-loop control system for both the displacement and attitude (Lab 4).

4.1 Laboratory preparation

1. Find an attitude controller from the PID family such that the closed-loop system follows, as much as possible, the specifications:

   - Zero steady-state error for a step reference
   - Zero steady-state error for a ramp reference
   - Settling time $T_s(2\%) < 3s$
   - Percentage Overshoot less than 10%

   You can use the root locus to guide in the design of the controller (see MATLAB's `rlocus` or `sisotool`)

2. Notice that block `PID Controller` from Simulink's library `Continuous` requires gains P, I and D, as in

$$G_c(s) = P + \frac{I}{s} + Ds, \tag{4.1}$$

and not the parameters K_p, T_i and T_d, as in

$$G_c(s) = K_p \left(1 + \frac{1}{T_i\, s} + T_d\, s\right). \tag{4.2}$$

3. Check the performance of the attitude controller using your Simulink simulation considering different θ_{ref}: a step, a ramp, and a parabola. Comment your forward/longitudinal model so that you only test the lateral motion.

   Plot:

   (a) The reference orientation $\theta_{ref}(t)$ vs the output $\theta(t)$
   (b) The tracking error $e_\theta(t) = \theta_{ref}(t) - \theta(t)$
   (c) The control action command, $\Delta\beta^{com}(t)$ vs the saturated input, $\Delta\beta(t)$.

After designing a closed-loop controller for the attitude, uncomment the longitudinal/forward model and provide a ramp reference for d_ref so that the Rasteirinho will maintain a constant velocity. Then:

1. Design a reference d_{ref} such that Rasteirinho follows a circular path.
2. Design a reference d_{ref} such that Rasteirinho follows a square path.
3. Design a reference d_{ref} that you would like to see the Rasteirinho following.
4. Test your longitudinal and lateral controllers for displacement and attitude using the previously defined trajectories. Plot:

   (a) The reference displacement $d_{ref}(t)$ vs the measured displacement $d(t)$ and the reference orientation $\theta_{ref}(t)$ vs the measured orientation $\theta(t)$
   (b) The displacement tracking error $e_d(t) = d_{ref}(t) - d(t)$ and the orientation tracking error $e_\theta(t) = \theta_{ref}(t) - \theta(t)$
   (c) The control action commands, $(\bar{\beta}^{com}(t)$, $\Delta\beta^{com}(t))$ vs the saturated inputs, $(\bar{\beta}(t)$,$\Delta\beta(t))$. Do not forget to verify if actuator saturation occurs. Remember that when you control both displacement and attitude in simulation, you must compute saturation as described in Figure 2.3.

5. Copy the appropriate parts you added to your Rasteirinho model in this section to the Simulink file that controls the real Rasteirinho, to test the closed-loop controllers in the laboratory.

4.2 In the laboratory

1. Set a constant velocity for d_{ref} and test the Rasteirinho attitude controller for the step, ramp, and parabola references you tested in simulation, and verify its performance.
2. Test both attitude and displacement controllers with the circular and square trajectories you designed. Verify if the trajectories performed correspond to the desired path, as measured by odometry, but also visually (does the Rasteirinho actually closes the square/circle? Does it ends in the same place where it started?)
3. Test both attitude and displacement controllers in the third (custom) trajectory you designed. Verify if the trajectory performed corresponds to the path you had envisioned.

Chapter 5

Controlling a mobile robot with feedback from optical tracking system

The objective of this Chapter is to control the Rasteirinho MIMO plant in closed-loop, with cartesian position and orientation feedback given by an optical tracking system (an indoor GPS), instead of using the odometry. The optical tracking system measures the position ${}^{W}r = (x, y)$ and attitude θ of the Rasteirinho in the Arena reference frame W. The goal is to control the Rasteirinho through predefined cartesian space trajectories $(x_{ref}(t), y_{ref}(t), \theta_{ref}(t))$.

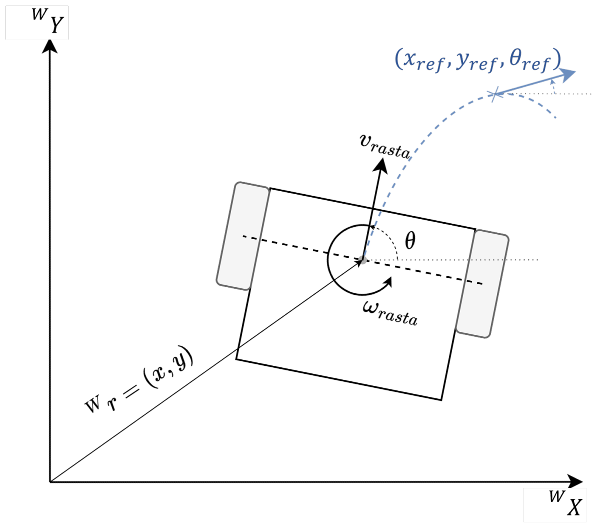


Figure 5.1: Cartesian space coordinates of the Rasteirinho in the Optical Tracking System reference frame.

We first need to determine the expression that describes the cartesian space trajectory ${}^{W}r(t) = (x(t), y(t))$ executed by the Rasteirinho (including attitude) from its linear velocity

$$v_{rasta}(t) = \alpha_d \bar{\beta}(t) \tag{5.1}$$

and the attitude $\theta(t)$, starting at initial position (x_0, y_0), with initial attitude θ_0.

First, we project the linear velocity of the robot onto the cartesian reference frame $({}^{W}X, {}^{W}Y)$, to get the velocity of the Rasterinho expressed in the cartesian space, $(\dot{x}, \dot{y})$:

$$\dot{x} = v_{rasta} \cos\theta \tag{5.2}$$

$$\dot{y} = v_{rasta} \sin\theta \tag{5.3}$$

Then, the position of the robot ${}^{W}r(t) = (x(t), y(t))$ is computed by integration of the velocity $(\dot{x}, \dot{y})$:

$$x(t) = x_0 + \int_0^t \dot{x}(\tau) d\tau \tag{5.4}$$

$$y(t) = y_0 + \int_0^t \dot{y}(\tau) d\tau \tag{5.5}$$

Consider a point $(x_{ref}(t), y_{ref}(t), \theta_{ref}(t))$ in the reference trajectory and the current cartesian position and orientation of the Rasteirinho (x, y, θ). The displacement error can be taken as the minimum distance (signed, $\sigma = \pm 1$) between the current position of the robot and the reference position at time t (5.6). The attitude error e_θ is given by the difference between the current and the reference orientation (5.7).

$$e_d(t) = \sigma \sqrt{(x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2} \tag{5.6}$$

$$e_\theta(t) = \theta_{ref}(t) - \theta(t) \tag{5.7}$$

The sign σ of the error distance depends on the orientation of the robot and on the location of the reference cartesian position. Simply put, if the robot is moving towards the reference position, the sign is positive; otherwise, if the robot is moving away from the reference position, the sign is positive. This is the same as taking the sign of the projection of the distance error vector $\overrightarrow{e_d} = (x_{ref} - x, y_{ref} - y)$ onto the current direction vector, $\overrightarrow{u} = (\cos\theta, \sin\theta)$. Hence, $\sigma = \text{sign}(\overrightarrow{e_d}.\overrightarrow{u})$.

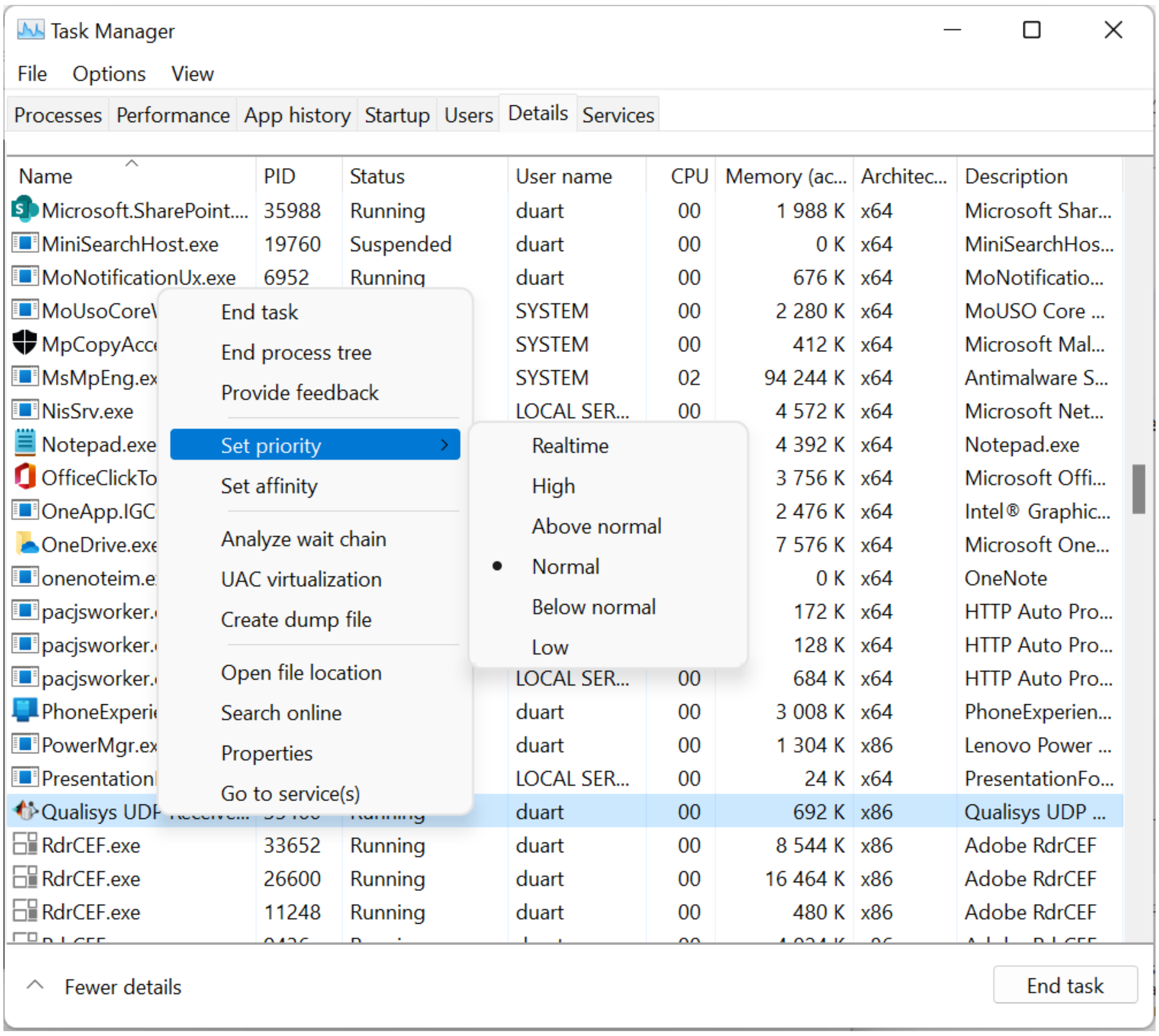
5.1 Laboratory preparation

1. Adapt your previous simulation file, with the Rasteirinho simulation model, and implement the expressions Eqs. (5.1)-(5.5) that describe the cartesian space trajectory ${}^{W}r(t) = (x(t), y(t))$ executed by the Rasteirinho. Use the integrator block where you can define the initial conditions x_0, y_0).
2. In the simulation file, implement the displacement and attitude error metrics in Eqs. (5.6) and (5.7). Use the Matlab `angDiff` function to compute the distance between two angles.
3. Define the reference pose (position and orientation) in cartesian space for the circular and square reference trajectories from the previous lab. Consider the initial position and orientation of the Rasteirinho to be zero in your trajectories.
4. Validate the performance of your controllers for the specified trajectories in simulation.

5.2 In the laboratory

For this laboratory, you will be using one of the computers in the lab. There is a Raterinho assigned to each computer.

1. Make sure the computer is connected to the TeamQuad Wifi Network.
2. Go to the folder corresponding to Lab 5. Run the program `QualysisUDPReceive.exe` (outside of MATLAB). To know if it is running properly, you should see new lines in the command window appearing continuously. Leave it running in the background during the class. Open the Task Manager, and in separator `Details` right-click `QualysisUDPReceive.exe` to set its priority to High; see Figure 5.2.
3. Run the initialization script.
4. Open the Simulink model `rastaWifi.slx`.



Name	PID	Status	User name	CPU	Memory (ac...	Architec...	Description
Microsoft.SharePoint....	35988	Running	duart	00	1 988 K	x64	Microsoft Shar...
MiniSearchHost.exe	19760	Suspended	duart	00	0 K	x64	MiniSearchHos...
MoNotificationUx.exe	6952	Running	duart	00	676 K	x64	MoNotificatio...
MoUsoCore\			SYSTEM	00	2 280 K	x64	MoUSO Core ...
MpCopyAcc			SYSTEM	00	412 K	x64	Microsoft Mal...
MsMpEng.ex			SYSTEM	02	94 244 K	x64	Antimalware S...
NisSrv.exe			LOCAL SER...	00	4 572 K	x64	Microsoft Net...
Notepad.exe					4 392 K	x64	Notepad.exe
OfficeClickTo					3 756 K	x64	Microsoft Offi...
OneApp.IGC					2 476 K	x64	Intel® Graphic...
OneDrive.exe					7 576 K	x64	Microsoft One...
onenoteim.e					0 K	x64	OneNote
pacjsworker.					172 K	x64	HTTP Auto Pro...
pacjsworker.					128 K	x64	HTTP Auto Pro...
pacjsworker.			LOCAL SER...	00	684 K	x64	HTTP Auto Pro...
PhoneExperi			duart	00	3 008 K	x64	PhoneExperien...
PowerMgr.ex			duart	00	1 304 K	x86	Lenovo Power ...
Presentation			LOCAL SER...	00	24 K	x64	PresentationFo...
Qualisys UDP			duart	00	692 K	x86	Qualisys UDP ...
RdrCEF.exe	33652	Running	duart	00	8 544 K	x86	Adobe RdrCEF
RdrCEF.exe	26600	Running	duart	00	16 464 K	x86	Adobe RdrCEF
RdrCEF.exe	11248	Running	duart	00	480 K	x86	Adobe RdrCEF

Figure 5.2: Changing the priority of `QualysisUDPReceive.exe` in the Task Manager.

5. Test the model with constant control action inputs to check that you can send commands to the Rasteirinho and read its pose from the optical tracking system (check the Scopes).

6. Now, send $\beta_L = \beta_R =$constant and verify that the Rasteirinho is actually moving forward only. Stop the movement. Check the Robot position measured from the optical tracking system. Since the reference frame of the markers placed on top of the Rasteirnho is not aligned with its axis of movement, when the robot is moving forward, the markers' reference frame will actually move in a straight line with a slope. Take the final cartesian coordinates measured (x, y) and compute the orientation offset: `orientOffset=atan2(y,x)`. Update the variable `orientOffset` in the initialization script given and rerun the script, so that the variable is updated in the workspace.

7. Repeat the previous test and verify that the executed trajectory in cartesian coordinates has only a component in x (i.e. $y = 0$).

8. Set your controllers for the attitude and displacement inside the initialization script, using the gain variables (which are then used in the PID-controllers inside the `Controllers` block). Validate it first on a simple straight-line trajectory. It should move forward only. Remember that you are still controlling the attitude, not just displacement.

9. Validate the performance of the controllers for the circle and the square trajectories, starting from $(0, 0)$. Plot in a 2D figure the executed trajectory and the reference. Also, plot the evolution with time of the position and the attitude errors. Show the results to the instructor.

10. Compare the trajectories measured by the optical tracking system with the odometry.