System Identification Laboratory Guide

Duarte Valério, João Oliveira, Rui Coelho

September 11, 2023

Contents

Introduction

> The reader may please to observe, that, in the last article of the recovery of my liberty, the emperor stipulates to allow me a quantity of meat and drink sufficient for the support of 1724 Lilliputians. Some time after, asking a friend at court how they came to fix on that determinate number, he told me that his majesty's mathematicians, having taken the height of my body by the help of a quadrant, and finding it to exceed theirs in the proportion of twelve to one, they concluded from the similarity of their bodies, that mine must contain at least 1724 of theirs, and consequently would require as much food as was necessary to support that number of Lilliputians. By which the reader may conceive an idea of the ingenuity of that people, as well as the prudent and exact economy of so great a prince.
>
> Jonathan Swift (1667 – †1745), *[Gulliver's] Travels into Several Remote Nations of the World*, I 3

There are five laboratory classes during the semester. Students must enrol in groups, and attend one of the available laboratory class shifts. Each group will work with a particular experiment during the semester.

In the current academic year's course webpage on Fenix, you can find:

- days of laboratory classes,
- rules on group formation,
- additional files needed to run some experiments,
- rules on class attendance (e.g. which classes are compulsory, penalties for arriving late or missing classes, how cases of force majeure are handled),
- whether the evaluation takes place by presentations in class or by delivering reports,
- particular rules for the evaluation of your laboratory work: how many presentations there will be (if any), in which dates they will take place, and their duration; or else how many reports should be delivered (if any), their deadlines, and what are the penalties for missing said deadlines.

Enrolment in groups uses Fenix as well; so does report delivery, if reports are due, or presentation delivery, should the presentations have to be delivered before class. To deliver a report, do the following:

- save your report as a pdf file;
- zip it together with
  - the files where you recorded experimental data,
  - the files with the code you used in identification;
- check if the size of the file exceeds the maximum that Fenix allows;
- upload the zip file to Fenix.

Acknowledgements

Professor Jorge Martins guided me through the practical details of how the experiments work, and provided a list of questions from former academic years which is to a great extent found in Chapter 2. Professor José Sá da Costa read the original version of this text from 2013, and provided useful comments. Mistakes and imperfections still found in this text are of course the author's.

Chapter 1

The experiments

> Sobre uma mesa, ao meio do pavilhão, estava assente um aparelho que eu nunca vira. Esse aparelho, em funcionamento, é que provocava o estranho ruído e, decerto, abrasava o ambiente. Era como que um pequeno motor cujo volante fosse substituído por uma hélice formada por um sistema de três ampolas de vidro. As ampolas continham uma substância roxa e *dardejavam em torno de si um halo de luz negra.*
>
> Mário de Sá-Carneiro (1890 — †1916), *A estranha morte do Professor Antena*

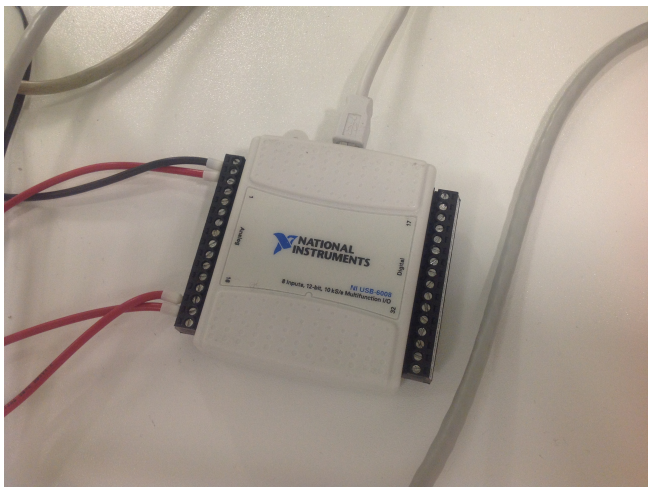
1.1 Generalities

The experiments described below in alphabetical order can be found in one of two laboratories:

- the Control Systems laboratory, which is part of the Control, Automation and Robotics Laboratory, in the basement of the Mechanical Engineering III building;
- the Electrical Machines Laboratory, in the ground floor of the Electrical Engineering building.

Check with the instructor which ones are available in the current semester, and in which shifts.

Many experiments in the Control Systems laboratory are controlled through a NI USB-6008 board connected to the USB port of a personal computer. The experiments that do not use this board are the ball on beam experiment (section 1.3) and the Rasteirinho (section 1.9).



The board has outputs (which will, of course, be the experiments' inputs) in the 0 V – 5 V range, and inputs (for the experiments' outputs) in the ±10 V range. The software used to control the experiments will be MATLAB. In several experiments, a real-time toolbox is not resorted to; hence do **not** move the mouse while the experiment is running, under pain of data communication failures.

Should the board freeze, do as follows:

- Look on the desktop for the icon of programme *Measurement and Automation*.
- Choose *Devices and Interfaces* > *NI-DAQmx* > *NI USB-6008*.
- Reset the board and request a self-test to check if it is now ok.

Also verify, in that programme, that the number of the board being used is 1 (otherwise, MATLAB will not recognise it). Change the name of the board to finish in 1 if it is not.

Computers in the laboratory either have no password (try hitting Return if asked for one), or have a password that your instructor will give you.

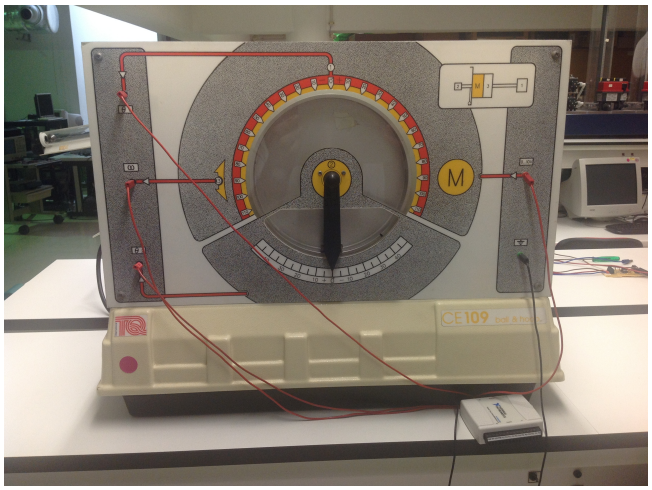
All SIMULINK files include a sum point to correct a possible zero offset from the sensors.

The solver is always set to `ode4`. Check if it is so in your case.

Save for the Rasteirinho, the sampling time should be $T_s = 0.01$ s in all cases, at least initially. (Obviously, processes differ, and it may be convenient to subsample data, or even change the sample time: the choice of a sampling time is actually one of the tasks you will have to perform.) It must be set both in `Simulation parameters` and in the `RTBlock`.

We will not convert sensors' readings into SI units (metre, radian, radian per second, Kelvin...), but rather work with the sensors' readings in V.

1.2 Ball in hoop



Where. Control Laboratory.

Description. A ball rotates on a hoop.

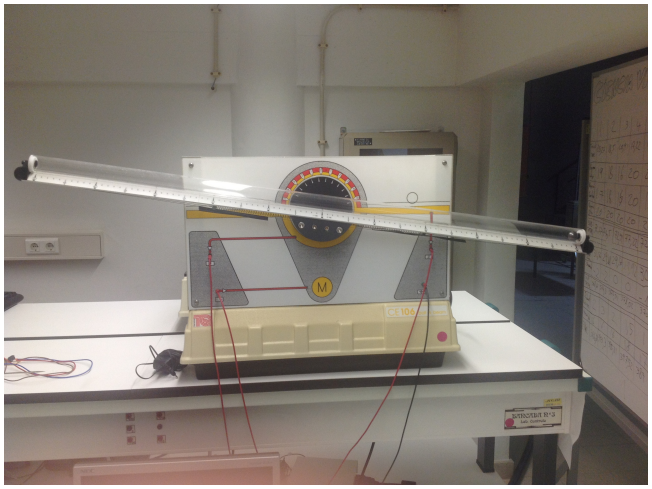
File. `Desktop\Ball in hoop\Ident_BallHoop.mdl`

Input. Signal in the 0 V – ±10 V range to rotate the hoop. We will use a signal in the 0 V – 5 V range only.

Outputs. 1. Position of the hoop θ_1. 2. Angular speed of the hoop ω. 3. Position of the pendulum θ_2.

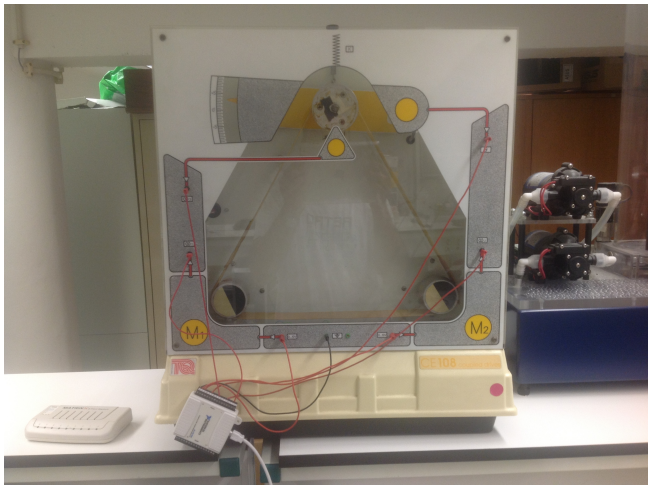
Remark. The ball must be in the hoop, within the pendulum, at all times.

1.3 Ball on beam



Where. Control Laboratory.
Description. A ball rotates sidewards on a beam, the slope of which can be controlled.
File. `Desktop\BBeam IS 1314\Ident_BallBeam.mdl`
Remark. This plant is unstable. It must be identified in closed loop. Although a proportional controller suffices when the output is the beam angle, given the eccentric behavior of the plant we add an integrator to the control loop, which improves the overall behavior of the closed loop plant. The open loop transfer function (the input of which is a signal to the motor that rotates the beam) is deduced therefrom. The beam is rotated with an eccentric, resulting in a rather nonlinear behaviour.
Outputs. 1. Beam angle θ, measured around the horizontal position (which is 0). 2. Ball position x, measured from the center. The latter quantity is measured from the resistance between the two wires on which the ball rotates, and which (being metallic) it short-circuits (we advise the students to avoid this reading, instead focusing on angle the beam as an output, because the sensor malfunctions regularly).

1.4 Coupled drives



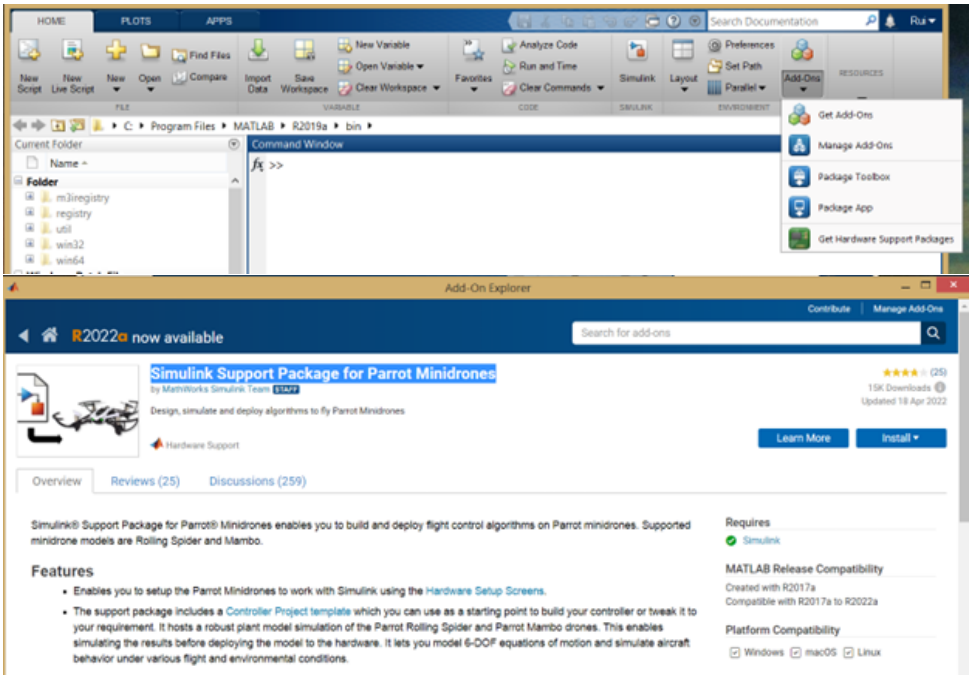
Where. Control Laboratory.
Description. Three pulleys, two of which connected to motors, rotate connected by a flat belt.
File. `Desktop\Motores acoplados\Ident.mdl`
Inputs. Two signals in the 0 V – ±10 V range to move the two motors of the two lower pulleys. In both cases, we will use signals in the 0 V – 5 V range only.
Variations. The upper pulley may be fixed (with a screw) or allowed to oscillate with a spring connected to an arm.

Outputs. Angular speeds of the three pulleys ω_1, ω_2 and ω_3. Angle θ of the oscillating arm of the upper pulley.

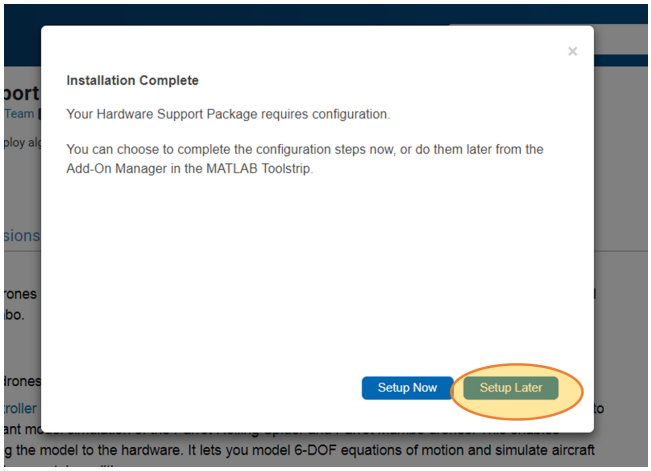
1.5 Drone

Where. Control Laboratory.
Description. The minidrone Parrot Mambo Fly flies up and down.
Set up. For this experiment, you must bring a laptop with MATLAB 2022b or higher and SIMULINK CODER, running on Windows. Before using the drone for the first time, open MATLAB, choose *Get Add-ons*, search for the *Simulink Support Package for Parrot Minidrones*, and install it. You also need the following toolboxes

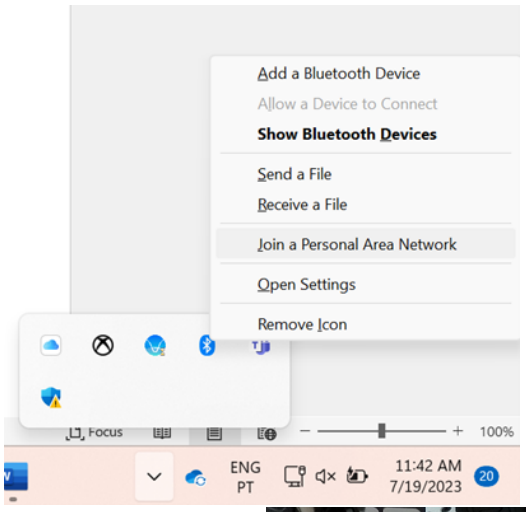
- *Robotics System Toolbox*
- *Aerospace Toolbox*
- *Aerospace Blockset*

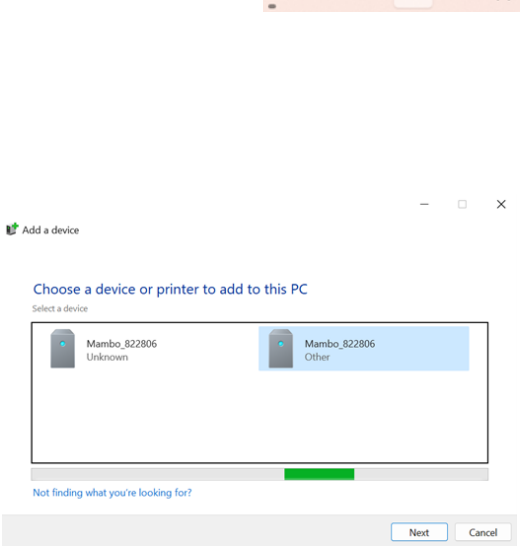


When installation is complete, press *Setup Later* (the drones have been previously configured, so there is no need to do anything else here).

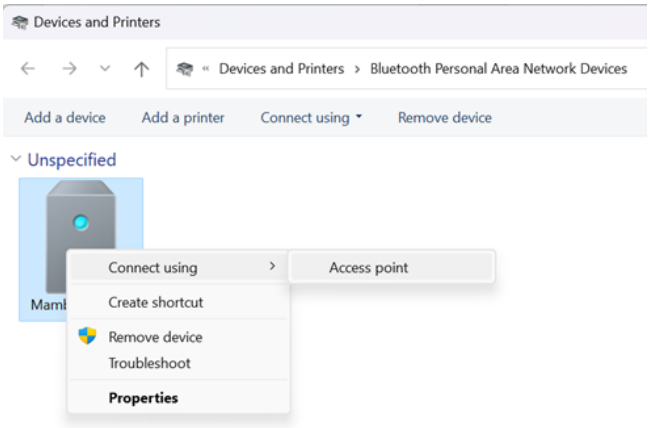


Insert a battery in the drone, and push the power button on its bottom. You will connect it to your laptop using Bluetooth. Go to the Bluetooth symbol on your laptop's Windows task tray and select *Join a Personal Area Network*. Click *Add a device* to search for your drone. You should see two devices named `Mambo_XXXXXX`, ending with the drone's serial number shown on its corresponding box. Select the device that says *Other* (do not select the *Unknown* device), and press *Next* to install it. If *Other* does not show, search again (it may take a few attempts until it shows). You only need to do this once, since the device will be stored on the list of devices of your computer.

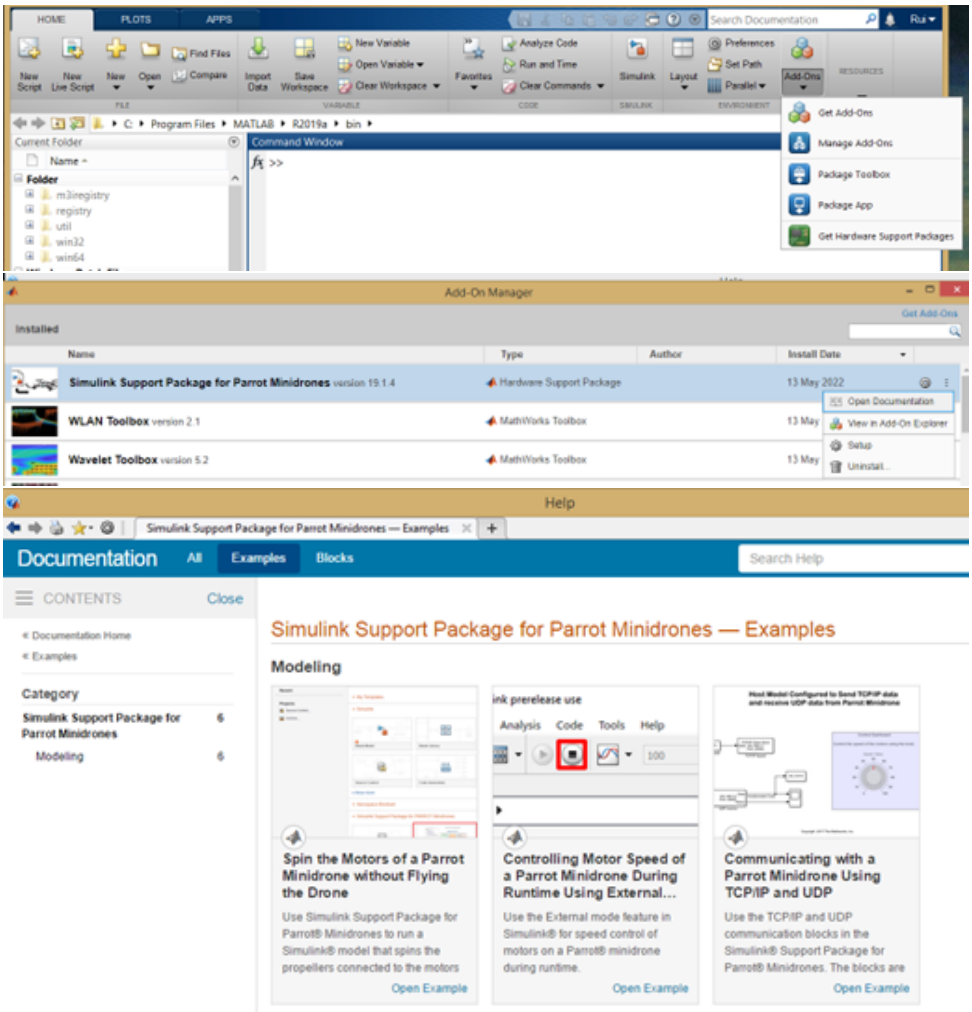




Right click the `Mambo_XXXXXX` you installed and select *Connect using* > *Access point*.

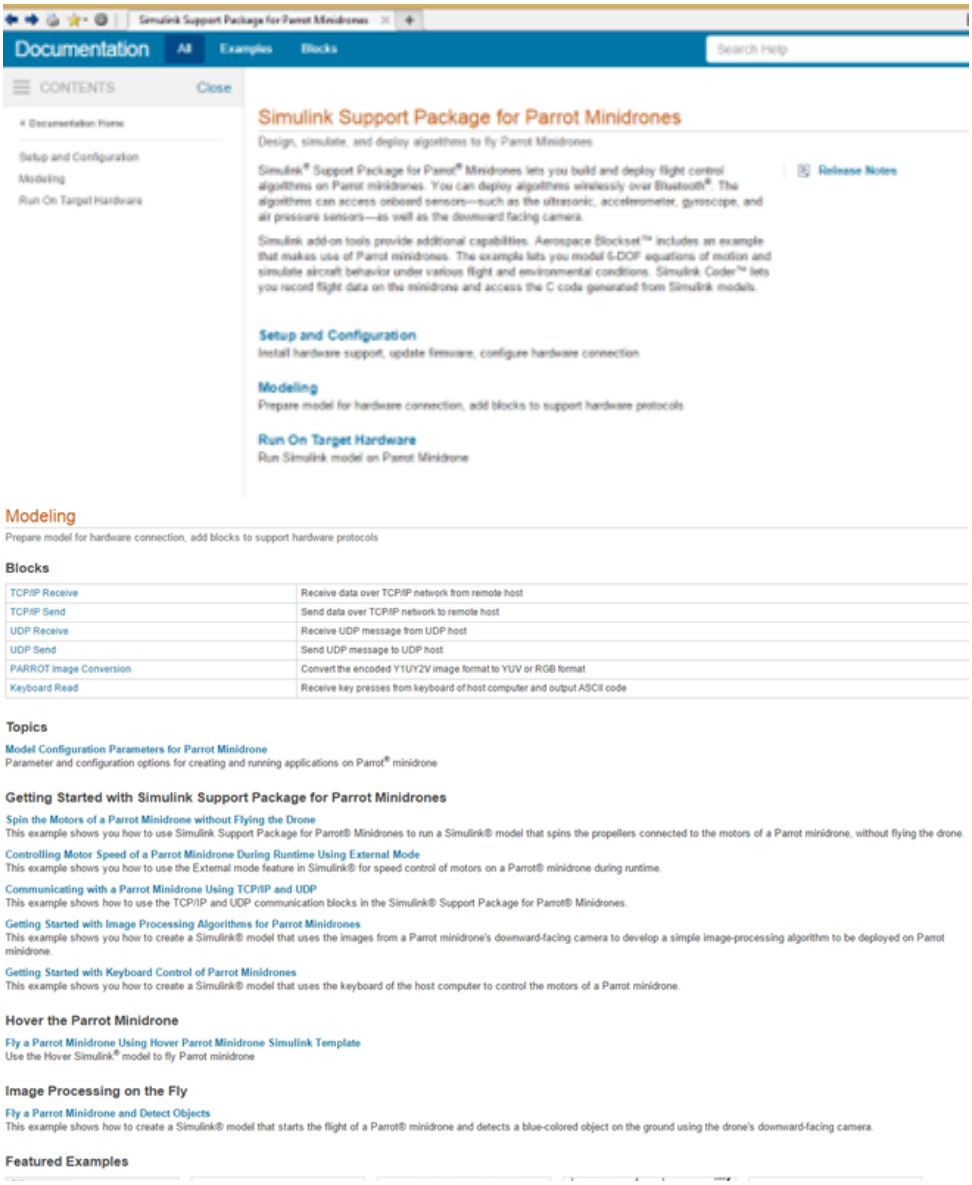


From MATLAB's Add-on Manager, open the support package documentation, and press *All*.

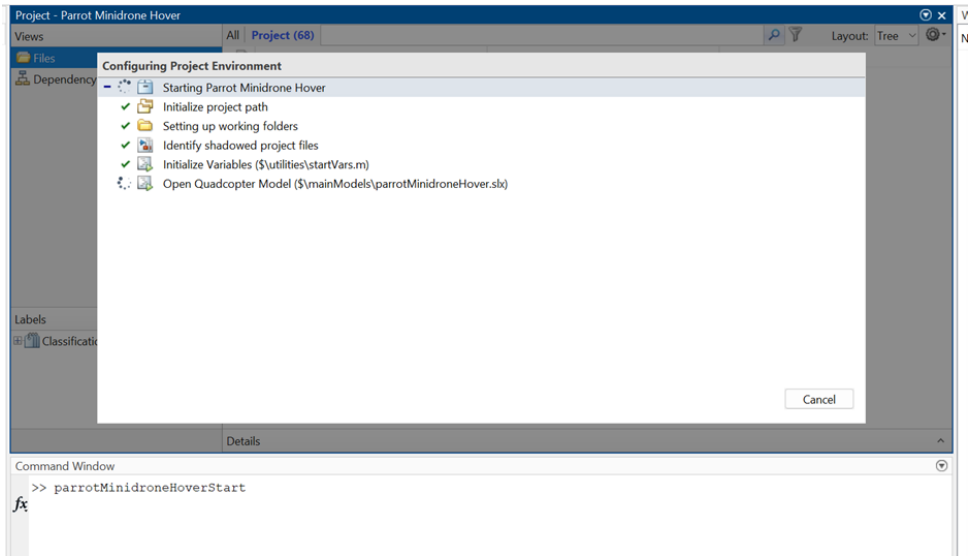


Go to *Modelling* and test the following examples (in this sequence):

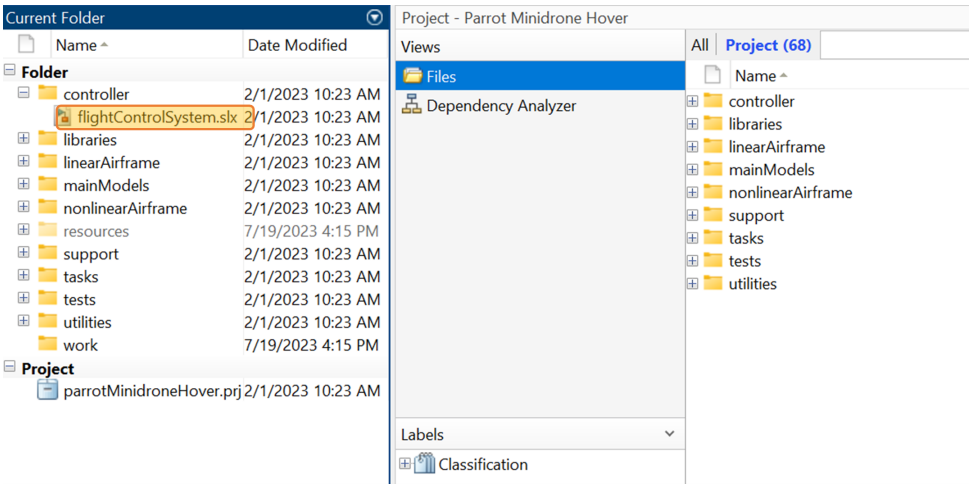
- Spin the Motors of a Parrot Minidrone without Flying the Drone;
- Controlling Motor Speed of a Parrot Minidrone during Runtime using External Mode;
- Hover the Parrot Minidrone.



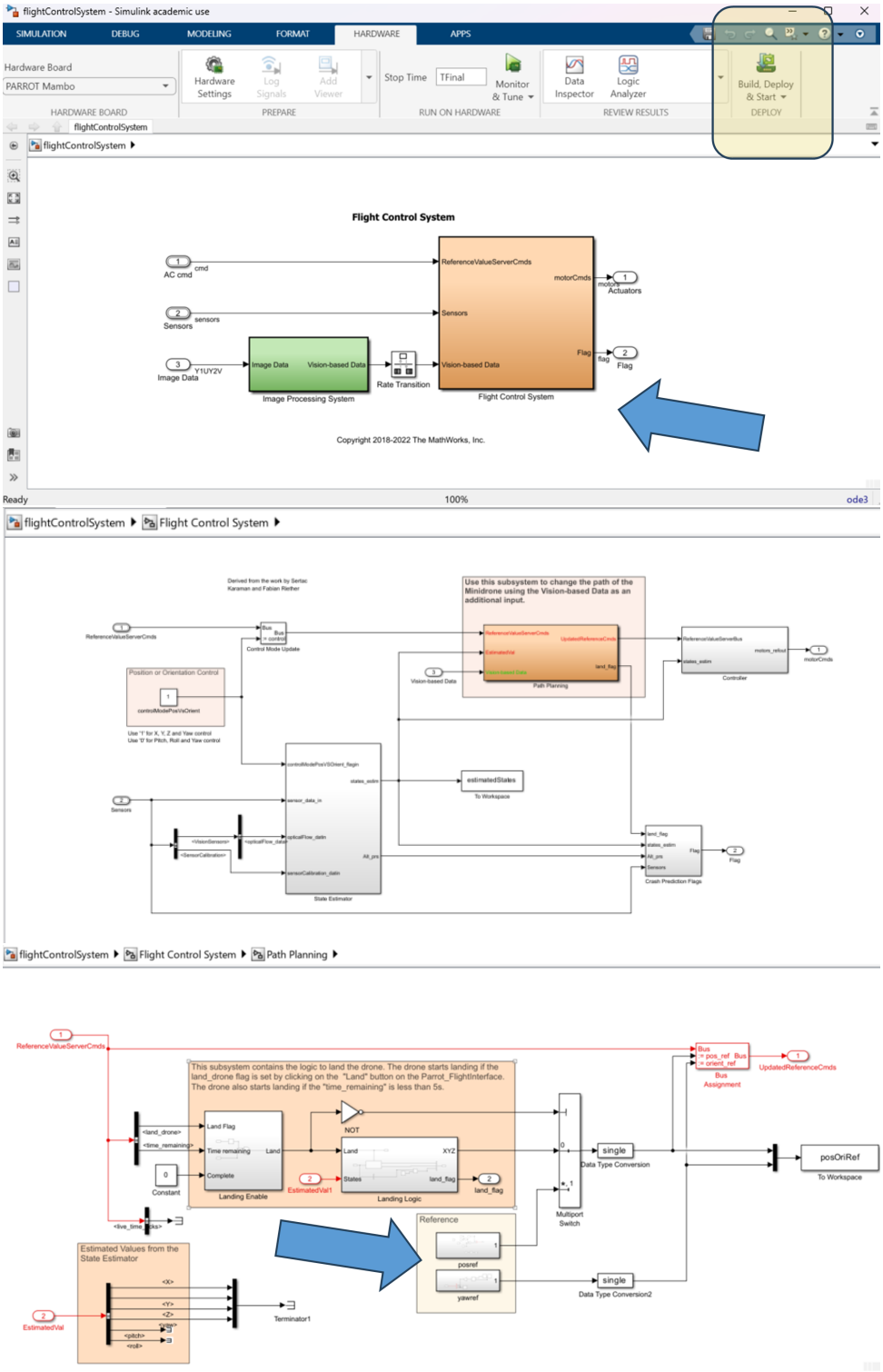
How to. Go to the Bluetooth symbol on your laptop's Windows task tray and select *Join a Personal Area Network*. Right click the `Mambo_XXXXXX` you installed as above and select *Connect using* > *Access point*. Type `parrotMinidroneHoverStart` on MATLAB's command window, to create a new project and launch it.



Replace file `flightControlSystem.slx` in folder *controller* by the file with the same name in the course webpage.

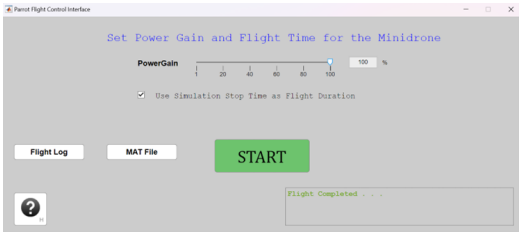


Open file `flightControlSystem.slx`, where the sensor processing and controllers are implemented, and that will be compiled and deployed on the drone's hardware when you click on *Build, Deploy and Start*. In the block Flight Control System, there is a block called *Path Planning*, in which you can change the references given to the controllers: cartesian position X,Y,Z and yaw (heading, rotation around Z-axis).



Press *Build, Deploy and Start*. When compilation and deployment is completed, an interface will open to command the start and stop of the flight. Set the gain value to 100%. Make sure your drone is on a level surface, with some pattern on the floor visible to the cameras. Press *Start*. The drone will follow the reference as prescribed. Press *Land* to slowly land the drone and *Stop* to turn off the motors immediately.

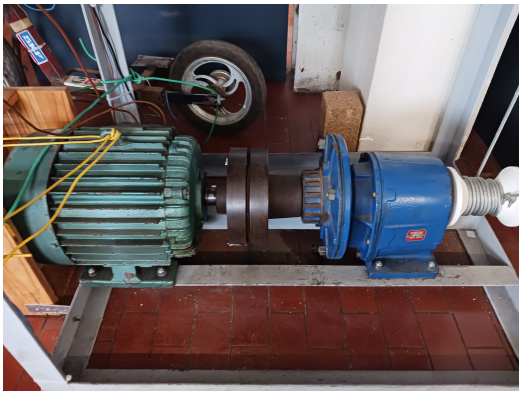
When the flight is completed, click on *MAT file* to download data recorded on the drone. This will create a file called `RSdata.mat` on your working folder. You can use the script `readDroneOnBoardData.m` to load the data from the file and plot the logged variables.



Input. Vertical Thrust.
Output. Height.

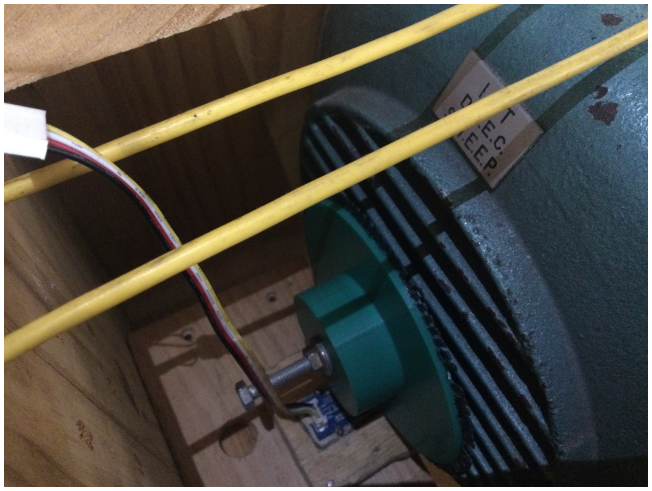
1.6 Elevator

Where. Electrical Machines Laboratory.
Description. An induction motor lifts and lowers a 20 kg mass (connected by a cable, through a gear box with a 24:1 reduction ratio, and a pulley).



Input. The motor is controlled with a V/f (voltage/frequency) algorithm, and thus the input signal is a frequency in Hertz. Thus, in steady state, the rotation speed of the motor will be directly proportional to the input.

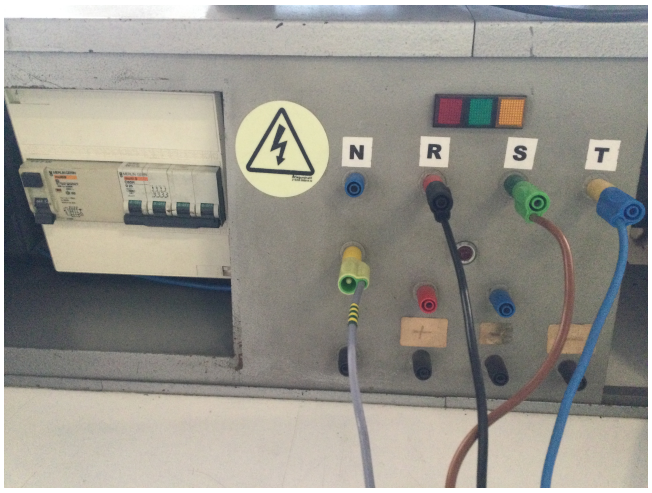
Output. Position (and velocity) of the mass can be obtained through an encoder (connected to an Arduino Due), with two sensors detecting the passage of the teeth of a cog, that measures the angular position (and velocity) of the shaft.



File. For this experiment, you must bring a laptop with MATLAB, and use the files available for this experiment in the course webpage. You must also install beforehand the following toolboxes:

- *Matlab Support Package for Arduino Hardware*;
- *Industrial Communication Toolbox*.

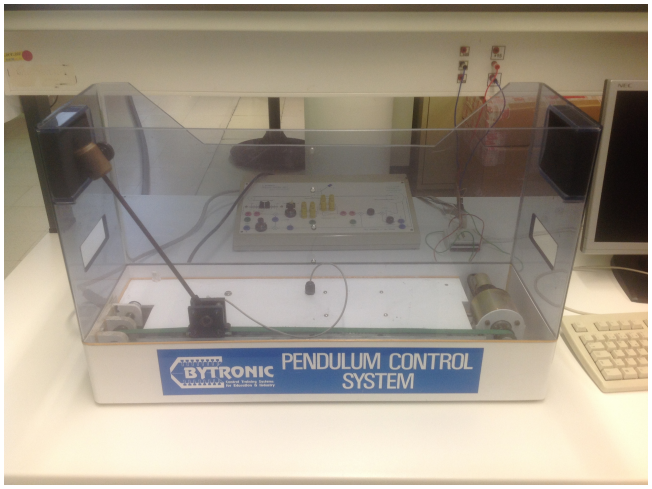
On/off. Connect the wires as shown below on the right, and *then* turn on the circuit breaker on the left. To turn off the elevator, first turn off the circuit breaker, and *then* unplug the cables.



How to. The frequency can be set manually in the controller below with the dial; turning it to the + side the weight goes up, and turning it to the − side the weight goes down. The motor starts when the green button is pressed, and stops with the red button. Use manual control to lift the weight above the wooden boards that protect the motor from the mass.

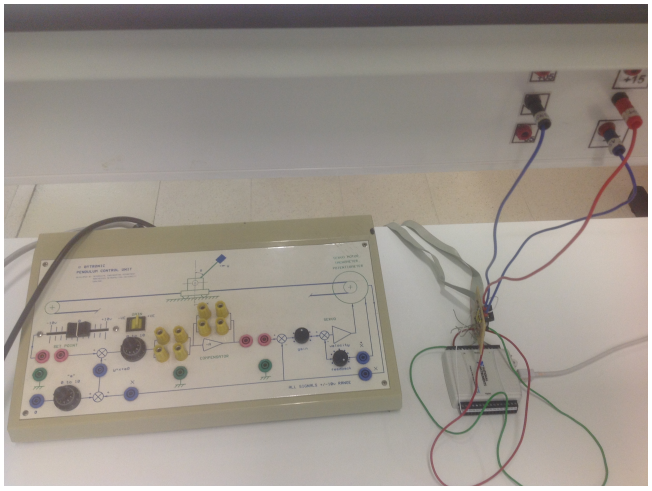
Use USB cables to connect your laptop to the Arduino Due (outputs) and the controller (inputs). When running the file provided, the frequency can no longer be set manually, neither does the motor start with the green button; the red button still works as a safeguard.

1.7 Pendulum



Where. Control Laboratory.
Description. A pendulum oscillates hinged on a base that moves sidewards.
Remark. The device was conceived to illustrate an *inverted* pendulum. We put it upside down because this way the system is stable. (Otherwise, it would be necessary to identify a closed loop, with a stabilising controller, and deduce therefrom the open loop.)

File. `Desktop\Ident Pendulo\Ident_Pend.mdl` (This file should be used together with MATLAB 2011a.)
Input. Signal to move the base of the pendulum sidewards. A board with an OpAmp conditions the signal in the 0 V – 5 V, duplicating it so that the full 0 V – 10 V range is used. The OpAmp must be fed.



Controller. The hardware provides a PD controller. This controller is needed to put the mass in the desired position. Setting the P button to 50% and the D button to 100% should provide a reasonable control action.

Settings. The mass of the pendulum can be moved along the shaft with a screw. We will use it as near the tip as possible. The button through which this position is set (for calculation purposes) must then be set to 2.7. (A wrong setting will always bring problems: a value too high will result in a non-minimum phase system; a value too low will make the measured movement faster than the reality.)

Outputs. 1. Position of the pendulum mass y, measured along a straight line from the middle of the box. 2. Position of the base of the pendulum x, measured from the middle of the box. 3. Angle of the pendulum θ. 4. Velocity of the base of the pendulum $\mathrm{d}x$.

1.8 Process trainer

Where. Control Laboratory.

Description. Water flows in two circuits, one heated and one refrigerated, connected by a heat exchanger.

File. `Desktop\Process Trainer\Ident_ProcessTrainer.mdl`

Input. 1. Signal to the pump of the heated circuit. 2. Signal to the heater of the heated circuit. 3. Signal to the pump of the refrigerated circuit. 4. Signal to a valve controlling the flow in the refrigerated circuit.

Variations. The following elements in the plant can only be controlled manually: the mixer in the heat exchanger (on/off); the valve that lets water flow from the heat exchanger to the reservoir of the refrigerated circuit (manual valve below the heat exchanger); the valve that opens the heat exchanger to the atmosphere (manual valve above the heat exchanger).

Output. 1. Flow in the heated circuit (FT1). 2. Temperature in the heated circuit after the heat exchanger (TT2). 3. Temperature in the reservoir of the heated circuit (TT1). 4. Temperature in the refrigerated circuit before the cooler (TT3). 5. Temperature in the refrigerated circuit after the cooler (TT4). 6. Flow in the

refrigerated circuit (FT2). 7. Air pressure in the refrigerated circuit within the heat exchanger (PT). 8. Water level in the refrigerated circuit within the heat exchanger (LT). 9. Temperature in the refrigerated circuit within the heat exchanger (TT5).

Remarks. 1. Most inputs can be controlled either through the NI USB-6008 board or manually. Check that they are being controlled the way that suits you. 2. Thermal processes involved are slow and will be avoided; so the flow in the heated circuit will be controlled using the circuit's pump, with all heaters off. (The refrigerated circuit might well be used instead.) 3. However, should a temperature be sought as controlled variable, two inputs (say, the heater and the pump of the refrigerated circuit) must be switched on manually and left at a constant value.

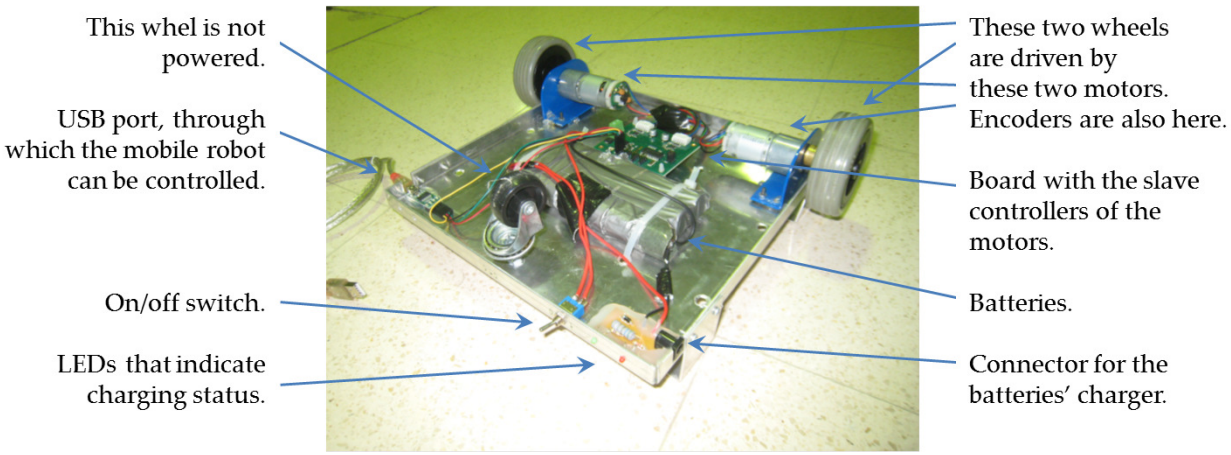
1.9 Rasteirinho

Where. Control Laboratory.

Description. A mobile robot follows a black line on the floor, controlled by a laptop.

Files. Download them from this link; use the version of the `slx` or `mdl` file that better suits your MATLAB version.

Inputs. The laptop, through the USB port, sends two 8-bit signals β_l and β_r to control the two motors of the two front wheels (left and right) of the Rasteirinho: 128 stops the motor, 255 makes it run forward at maximum speed, 0 makes it run backwards at maximum speed. We will use a constant, positive average value $\bar{\beta} = \frac{\beta_l+\beta_r}{2} - 128$ to make the Rasteirinho advance at constant speed, and vary the value of $\Delta\beta = \frac{\beta_r-\beta_l}{2}$ to make it turn left or right.



Outputs. The USB port is also used to read the values p_l and p_r of the encoders connected to each wheel, which provide 360 pulses per wheel revolution (so, assuming that the wheels do not slip, the distance the Rasteirinho moves forward along the line is given by $\frac{\Delta p_l+\Delta p_r}{2} \times \frac{2\pi r}{360}$, where r is the radius of the wheels). The

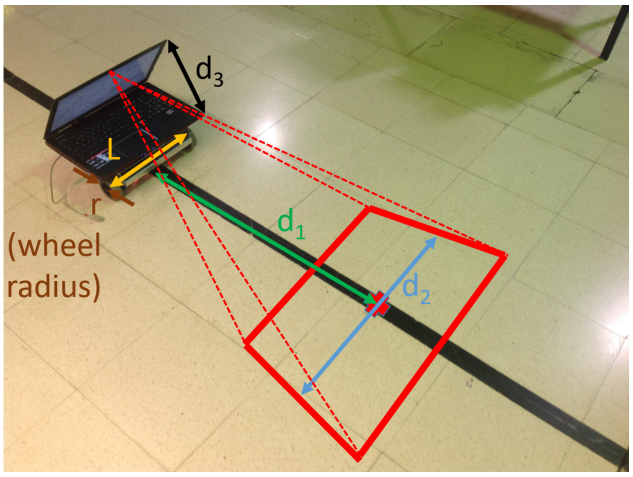
laptop's camera, with r rows and c columns, shows if the line is followed; it should be set for the lowest possible resolution, which is enough and saves processing time.

Controller. A SIMULINK file finds the coordinates x and y of the centroid of the black part of the image captured by the camera. The horizontal coordinate x should be $\frac{c}{2}$ (e.g. 80 for an image with 160×120 pixels). A PID controller finds from error $\xi = x - \frac{c}{2}$ a control action $\Delta\beta$.

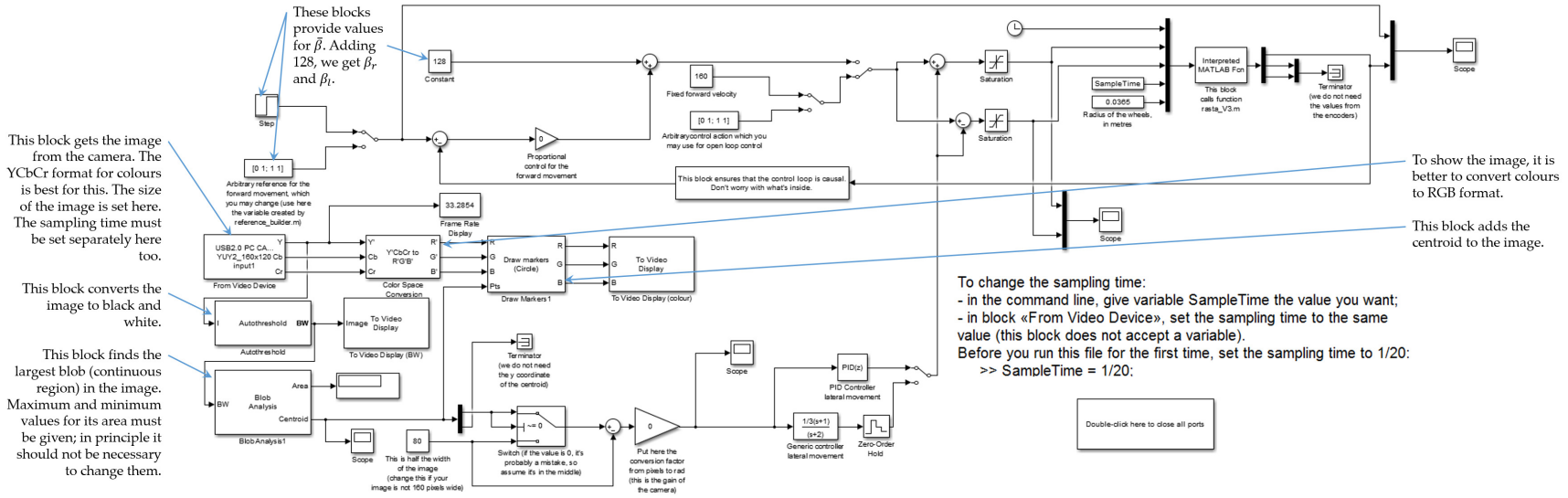
How to. Connect the USB cable. Verify which COM port is in use in `Control Panel > Device Manager`. Turn the Rasteirinho is on (a red LED should now be lit in the integrated circuit, other than the LED which is on when the Rasteirinho is charging). Open the SIMULINK model `Rcontrol.slx` (or `.mdl`). This SIMULINK file makes use of the code in `rasta_V3.m`, so open that file too and set the correct COM port. Using MATLAB's command prompt in the command window, set variable `SampleTime` to 1/20 (mind the upper and lower case letters). Set the correct value of the radius of the wheels of the Rasteirinho in file `Rcontrol.slx`. If the image of your camera is not 160 pixels wide, correct the value of the corresponding block. Leave $\bar{\beta}$ and $\Delta\beta$ equal to 0 (those are the default values), so that the Rasteirinho does not move, run the SIMULINK file, and open the block `To Video Display (colour)`. You should see the red mark showing the centre of the track on the middle of the screen. Check the actual frame rate display you can get with your laptop in block "Frame Rate Display". (It will, hopefully, be a more or less constant value.) We will adjust the fixed-step of the solver to the inverse of this value, thus ensuring (manually) that the file will run (approximately) in real time. Stop the simulation, and then change the sample time twice: change variable `SampleTime` using the command prompt, and change it again in block `From video device` (which for some reason does not accept variables as the sample time). Check that your file is indeed running approximately in real time, comparing the simulation time that SIMULINK shows when running with the values you get with a stop watch.

Remark. For instance, if you get a frame rate display of 25 fps (frames per second), set the sample time to 1/25 s; if it is 8 fps, the sample time will be 1/8 s.

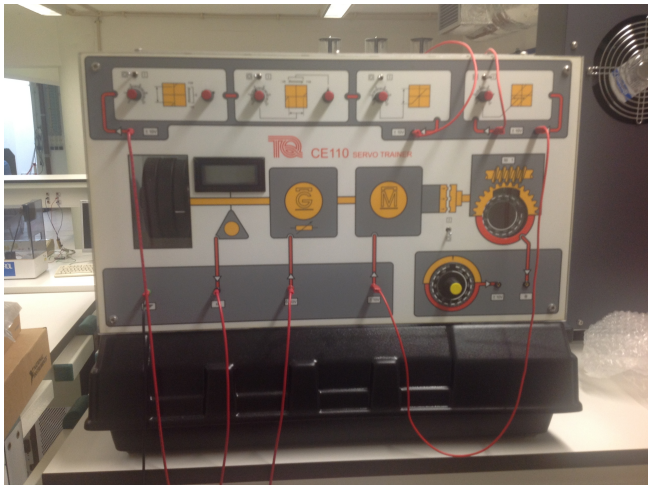
Variations. The Rasteirinho will be identified using $\Delta\beta$ as input and the attitude as output. Find the conversion factor $\frac{d_2}{c\,d_1}$ between ξ in pixels and the attitude in radians, as shown below, and put the value in the corresponding block. Add a disturbance to the output of the PID controller and identify a model from that input, using a constant value for $\bar{\beta}$.



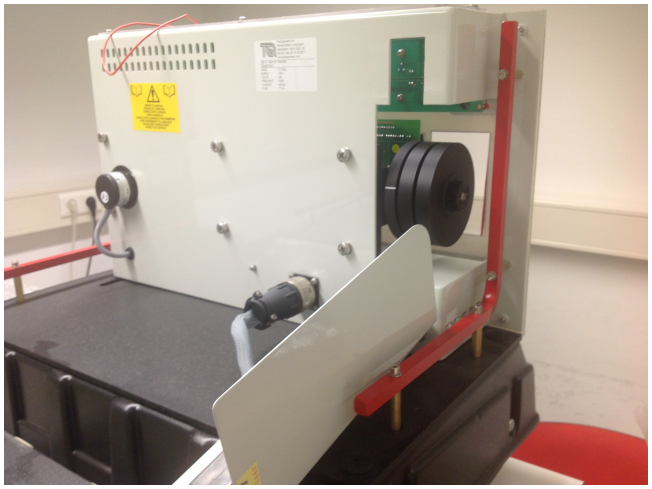
Hint. For further information on the Rasteirinho, you may want to consult the 2020 Laboratory Guide of the Control Systems course, which includes detailed instructions and a troubleshooting list.


These blocks provide values for $\vec{\beta}$. Adding 128, we get β_r and β_l.
This block gets the image from the camera. The YCbCr format for colours is best for this. The size of the image is set here. The sampling time must be set separately here too.
This block converts the image to black and white.
This block finds the largest blob (continuous region) in the image. Maximum and minimum values for its area must be given; in principle it should not be necessary to change them.
To show the image, it is better to convert colours to RGB format.
This block adds the centroid to the image.
To change the sampling time:
- in the command line, give variable SampleTime the value you want;
- in block «From Video Device», set the sampling time to the same value (this block does not accept a variable).
Before you run this file for the first time, set the sampling time to 1/20:
>> SampleTime = 1/20;
Double-click here to close all ports
Constant
Fixed forward velocity
Arbitrary control action which you may use for open loop control
Proportional control for the forward movement
This block ensures that the control loop is causal. Don't worry with what's inside.
Saturation
SampleTime
0.0365
Radius of the wheels, in metres
Interpreted MATLAB Fcn
This block calls function rasta_V3.m
Terminator (we do not need the values from the encoders)
Scope
Step
Arbitrary reference for the forward movement, which you may change (use here the variable created by reference_builder.m)
Frame Rate Display
From Video Device
Color Space Conversion
Draw Markers 1
To Video Display (colour)
Autothreshold
To Video Display (BW)
Blob Analysis1
Terminator (we do not need the y coordinate of the centroid)
This is half the width of the image (change this if your image is not 160 pixels wide)
Switch (if the value is 0, it's probably a mistake, so assume it's in the middle)
Put here the conversion factor from pixels to rad (this is the gain of the camera)
PID Controller lateral movement
Generic controller lateral movement
Zero-Order Hold

1.10 Servomotor

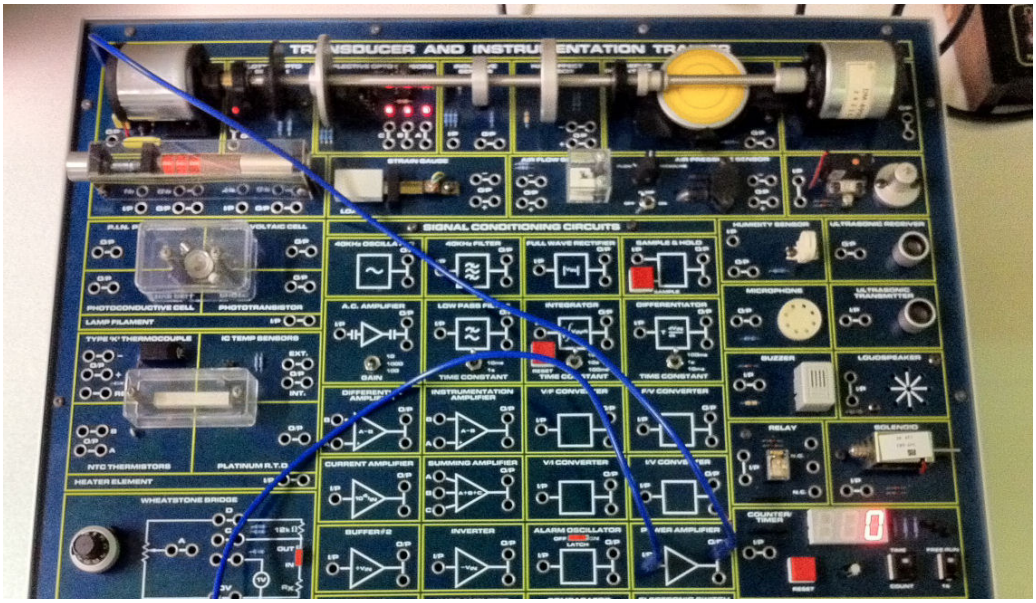


Where. Control Laboratory.
Description. A servomotor rotates a shaft with flywheels.
File. `Desktop\Servomotor\Servomotor.mdl`
Inputs. 1. Signal in the ± 10 V range to move the motor's rotor. A non-linearity can be added using the hardware. 2. Signal in the 0 V – 10 V range to engage a generator that breaks the rotation. In both cases, we will use signals in the 0 V – 5 V range only.
Variations. Flywheels may be added or removed (through an opening in the back) to change the system's inertia.



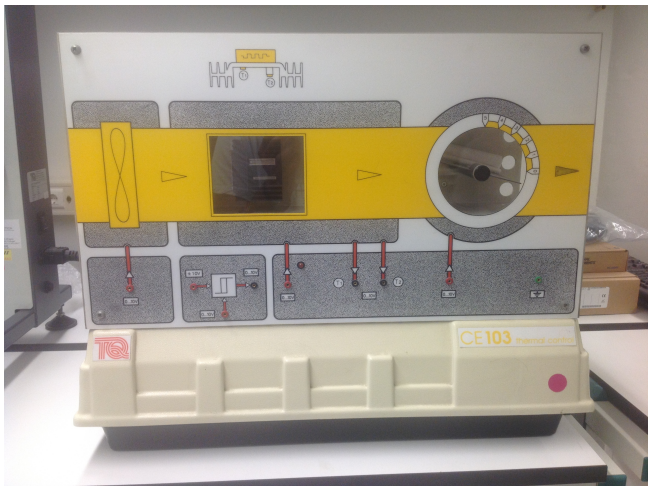
Output. Angular speed ω. (The angular position θ might be used instead, but this will not be done.)

1.11 Shaft rotation



Where. Control Laboratory.
Description. A shaft rotates moved by a DC motor. This is part of the DigiAC 1750 device.
File. `Desktop\System Identification - shaft\control.mdl`
Input. Signal in the ± 10 V range. (The motor could actually support the ± 12 V range.)
Remark. Since the NI USB-6008 board outputs a voltage in the 0 V – 5 V range, this is converted into the desired range by a signal conditioning circuit.
How to. Connect the output of the board to the input of the signal conditioning circuit. Also connect the ground of the board and the ground of the DigiAC to the ground of the signal conditioning. Then connect the ± 12 V of the DigiAC to the bus strips that feed the opamps in the signal conditioning circuit. Connect the output of the signal conditioning to the power amplifier (as shown in the figure above), and this to the DC motor.
Controller. The SIMULINK file implements a closed loop with proportional control.
Output. Position of the shaft.
How to. Connect the output of the potentiometer sensor (near the yellow roundel) to the input of the NI USB-6008 board. The resolution could be improved through a signal conditioning circuit, but is already excellent.

1.12 Thermal process



Where. Control Laboratory.

Description. Air is blown by a fan, heated, and passes through a butterfly valve.

File. `Desktop\Thermal\Ident_thermal.mdl`

Inputs. 1. A signal to rotate the fan. 2. A voltage to warm the air blown by the fan. 3. A signal to open or close the butterfly valve that changes the air flow. All signals are in the ± 10 V range, but we will use signals in the 0 V – 5 V range only.

Variations. We will leave the butterfly valve always open.

Outputs. 1. Temperature at the heating point T_1. 2. Temperature of the blown air T_2.

Remark. This is a rather slow process. Its time constant is very large. A step response must typically be recorded for 10 to 15 minutes to be of any use.

1.13 Viena suspension

Where. Electrical Machines Laboratory.

Description. The Viena (*Veículo Inteligente Elétrico de Navegação Autónoma*, i.e. Autonomous Navigation Intelligent Electric Vehicle) is a Fiat Seicento Elettra with new batteries and instrumentation. We will study the suspension of one of the front wheels.

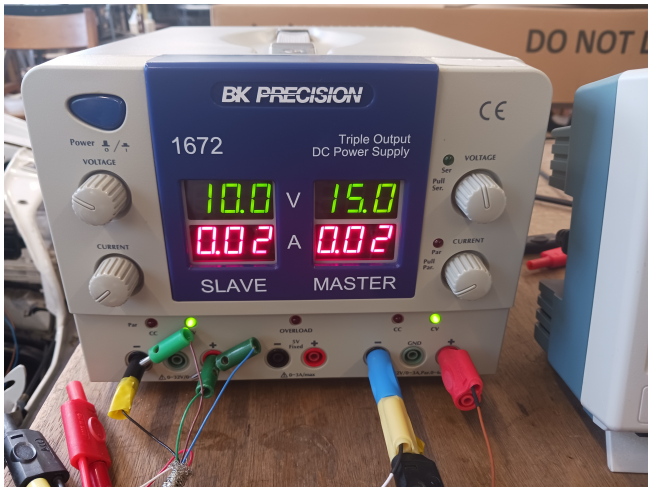
Input. A force manually exerted by one or more students on the chassis, measured by a force sensor.

Output. Distance of the chassis to the ground.

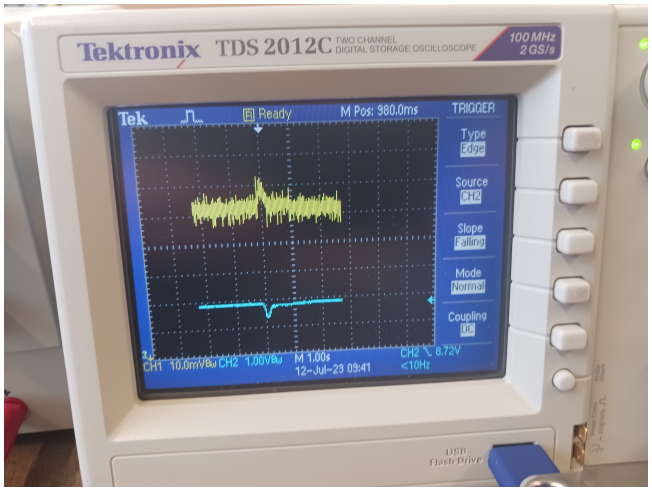
How to. The setup should always be prepared, if necessary, next to the class instructor. Rigidly attach the force sensor with two screws in the upper part of the car. Rigidly attach the ultrasound sensor to the front wheel of the car.

Due to range limitations of the sensor, make sure that you have the static support on the ground bellow the sensor, as shown in the following Figure.

Once the rig is ready, we must provide the power for both sensors to collect measurements by connecting it to the power supply shown in the following Figure.



Lastly we must connect an oscilloscope which takes the voltages from the sensors and convert it into a **cvs** file, where the oscilloscope is shown in the following Figure.



Remark Given that the actuation is imposed through human contact, some of the signals cannot be generated, i.e. coloured noise, pure sinusoids. . . ; thus other approaches must be employed, e.g. deconvolution.

Chapter 2

Laboratory work

— Nous en serons bien plus certains quand nous aurons essayé ! » répondit Pencroff.

Jules Verne (1828 — †1905), *L'île mistérieuse*, III 5

2.1 Classes 1 and 2

All groups in the class may agree upon a distribution of the systems. If this is not possible, systems will be distributed by drawing lots. Even in this case, you may swap your system with another group. Each group will keep working with the same system till the end of the semester.

This is the work to be carried out in these classes.

1. Choose among the input and output variables the ones you will work with. If your system can be configured in different manners, and no particular configuration is adopted in chapter 1, choose which one or which ones you will use. Discuss this with the instructor.

2. To the extent that you are able to do this for your system, find from the passing physical laws the differential equations that describe the dynamics of the system. Linearise them around a suitable point. Find the corresponding transfer function. Characterise any non-linear elements in the system. The importance of doing this at the same time that you collect the data is that, if you have an expected model, you can check if the data you are collecting is compatible with what you expect.

3. Choose a suitable sampling time for the system. If you find this really necessary, change the sampling time of your system as explained in section 1.1. Otherwise keep $T_s = 0.01$ s, and subsample the data as needed.

4. Record data for step responses. Use different amplitudes and see if the results agree with the plant being linear.

5. Record data for responses to sinusoids, so as to obtain the frequency response of your plant. Remember that you must wait for the steady-state response.

2.2 After classes 1 and 2

This is the work to be carried out outside class with the data recorded. Notice that not all items can be carried out for all systems and for all the sets of recorded data. If you are unable to do some of the items in the list below, explain why.

These tasks are previous to any system identification.

6. Choose different performance indexes to validate all the models you will find along the semester. Remember that, if your models do not all have the same structure, you should use the AIC and/or the BIC to weight performance improvement against the number of parameters in the model.

7. Process the data you collected, filtering it if it is noisy and if filtering improves the results obtained with the identification methods you will apply.

8. In relation with this, check if subsampling is necessary to increase the sampling time (i.e. decrease the sampling frequency).

9. Process the data you collected, to take into account that transfer functions presume zero initial conditions (i.e. make sure that initial conditions in the identification methods you will use are zero).

10. If you were able to find a model following the guidelines in point 2 above, verify your modelling options against the data found, and explain if this expected model influenced your choices of sampling time and of signals used for identification in the laboratory.

These tasks should follow every identification endeavour.

11. Check the performance of your models. Validate it with data you did not use for identification, including data with different inputs (e.g. when identifying from step responses, compare the identified model's responses to sinusoids with experimental responses to sinusoids). Provide suitable comments.

12. Compare all the models you obtained so far (including any model following the guidelines in point 2 above). Provide suitable comments. Justify the model orders provided to the identification models.

13. Check from the models whether the sampling time you chose was adequate or not.

These tasks concern identification from responses in time.

14. Identify the steady state of the system's step responses.

15. Identify the system from its step responses, using characteristics of the transient response.

16. Identify the system from responses in time using least squares.

These tasks concern identification from frequency responses.

17. Find your system's frequency response from steady-state responses to sinusoidal inputs.

18. Find your system's frequency response from the Fourier transform of the impulse response, found by deconvolution of some time responses.

19. Identify the system from the Bode diagram of its frequency response.

20. Identify the system using Levy's method.

These tasks concern real-time identification.

21. Implement an offline version of real-time identification using data from time responses, and using some previously found model as an initial estimate of the model's parameters. Apply a forgetting factor if you see fit.

22. Repeat this, if possible, for a zero initial estimate of the model's parameters.

2.3 Class 3

This is the work to be carried out in these classes.

23. Conceive reasonable random signals to feed your system with. Discuss them with the instructor.

24. Record data for responses to such signals.

2.4 After class 3

This is the work to be carried out outside class with the data recorded. The tasks previous to identification, and the tasks to be carried out after system identification, are the same.

These tasks concern continuous models.

25. Identify the system's static gain using the methods you learned for stochastic systems.
26. Identify a continuous model for your system from its impulse response, found as you learned for stochastic systems.
27. Identify a continuous model for your system from its frequency response, found as you learned for stochastic systems. Notice that you learned different ways of finding the frequency response for stochastic systems, and also different ways of identifying a model from a frequency response.
28. Find the coherence function and take conclusions therefrom.

These tasks concern discrete models.

25. Use the autocorrelation and the partial autocorrelation of the output to reach conclusions about model structure.
26. Identify MA and AR models for the plant, or, if this does not make sense, explain why.
27. Identify an ARMAX model for the plant, or, if this does not make sense, explain why.

2.5 Classes 4 and 5

In these classes, if you identified models that are not coherent, and have reason to suspect that this is because the data was improperly recorded or the experiences were carried out in poor conditions, you may repeat the experiments and collect new data.