Duino-Based Learning (DBL) in Control Engineering Courses

Eneko Lerma, Ramon Costa-Castelló, Robert Griñó
Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial (ESAII)
Universitat Politcnica de Catalunya (UPC)
eneko.lerma@estudiant.upc.edu, ramon.costa@upc.edu, roberto.grino@upc.edu

Carlos Sanchis
Mathworks
Carlos.Sanchis@mathworks.es

***Abstract*—This document presents a project to develop freely redistributable materials to conduct educational lab projects with MATLAB, Simulink, Arduino and low-cost plants. This work materials introduce the fundamentals of Control Engineering through exercises and videos. Along with all this, the most important steps and issues appeared in the project are explained, so anyone interested on doing a project can have a starting point instead of starting a project from scratch, which most of times this results hard to implement.**

I. Introduction

The process of reform of university studies, Bologna process, and technological advances have generated a revolution in the mechanisms used for teaching and teaching materials. Technological studies have been especially sensitive to these changes [1], [2].

Automatic control has been one of the most active areas in this reform. The improvements introduced are numerous. Some examples are interactive applications [3], [4], virtual laboratories [5], and remote laboratories [6]. Despite this, practical on-site experiences continue to be a fundamental element for any engineering study and in particular for control engineering [2].

The appearance of low-cost hardware such as the Arduino and the Rasperry-pi [7]–[10], together with the development of automatic code generation software [11] and the hardware in the loop technologies have facilitated much of the development of industrial applications and at the same time of environments of novel practices and attractive for students.

This paper describes a project to remodel of the teaching experimental laboratories from the automatic control laboratory of the School of Industrial Engineering (Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, ETSEIB) from the Technical University of Catalonia (Universitat Politècnica de Catalunya, UPC). In this laboratory students learn discrete-time control and automatic control [12]–[14]. The work describes how the laboratory has been modified to be able to use the new low-cost elements and the automatic generation of code.

The paper is organized as follows: Section II contains a description of previous and proposed experimental setup, Section III contains a description of the Duino-Based Learning, Section IV contains a couple of examples and finally Section V provides some conclusions and future works.

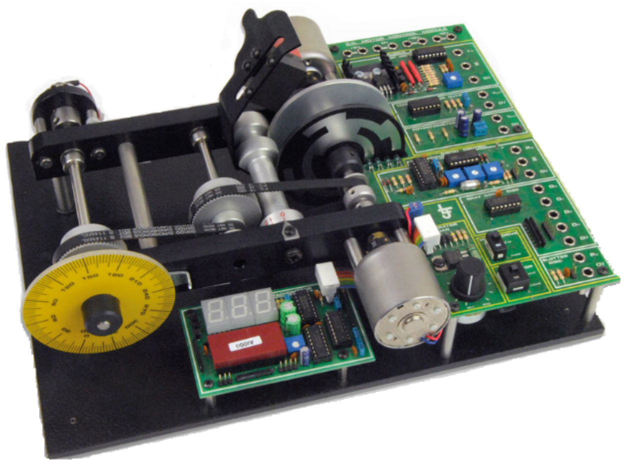


Fig. 1: LJ Technical Systems's servomotor.

II. Brief description of the experimental platform

A. Previous experimental setup

The experimental plant at the laboratory is a DC servo system from LJ Technical Systems, see Figure 1. This plant contains a DC motor along with its power driving electronics and signal conditioning stages. Specifically, the module contains the necessary sensing elements to close a speed control loop (tachometric dynamo and encoder) and a position control loop (potentiometer). In addition, the mechanical axis of the motor is equipped with a magnetic brake which can be used as a disturbance.

Before working with the new environment, students acquired speed and position data and send the control signal to the plant using an analog to digital/digital to analog (AD/DA) card built in the personal computer (PC), see Figure 2. This cards were very expensive and used the PCI bus of the PC which is becoming less and less common and, in this way, it complicates the continuous update and maintenance of the equipment.

It is worth noting that in this experimental platform the controller and its calculation were performed on the PC using a specific, and not standard, library for MatLab/Simulink.

B. Hardware issues

Recently, a decision was made to use an Arduino board for AD/DA conversion and controller code execution. The chosen

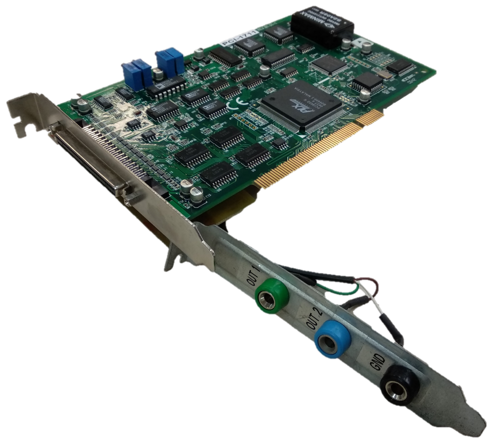


Fig. 2: AD/DA card with its conditioning stage.

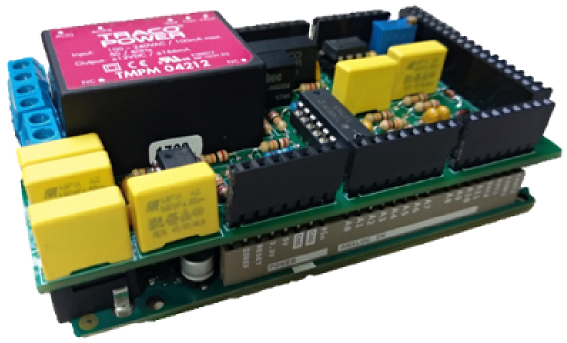
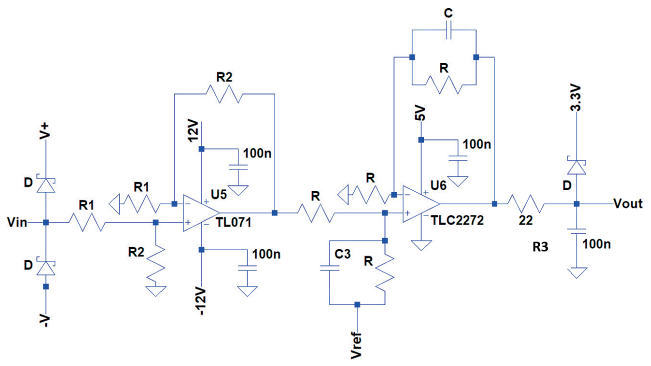


Fig. 3: Arduino Due signal conditioning board.

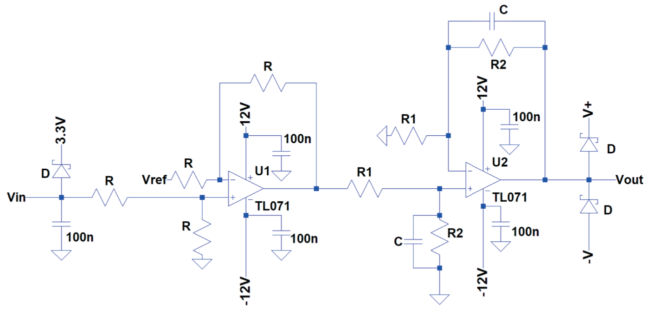
board was the Arduino Due board, due to three main reasons: it is one of the few Arduino boards equipped with two real DAC converter channels; it has twelve multiplexed analog input pins with three 12 bits AD converters, while other boards, such as the better known Arduino Uno or the Arduino Mega have only ten bits of resolution; and because the Arduino Due board has a high computing performance (ARM Cortex-M3 core) which improves the execution speed of a digital controllers. Another relevant reason, this of economic character, is that the Arduino board is much cheaper than the AD/DA conversion cards for PCI bus.

The auxiliary hardware also had to be modified as the input/output voltage ranges of both devices, the Arduino Due and the LJ servo-system plant, did not match. On one hand, the Arduino Due board has an operating voltage of 3.3 V. That is, the maximum voltage that the I/O pins can tolerate is 3.3 V and the minimum voltage is 0 V. On the other hand, the plant runs its inputs and outputs at ± 5 V. This meant that a conditioning hardware (signal adapter) had to be designed to connect the Arduino Due board to the DC servo system. As it can be seen on Figure 3 the conditioning hardware board was designed to be connected to the Arduino board as a shield.

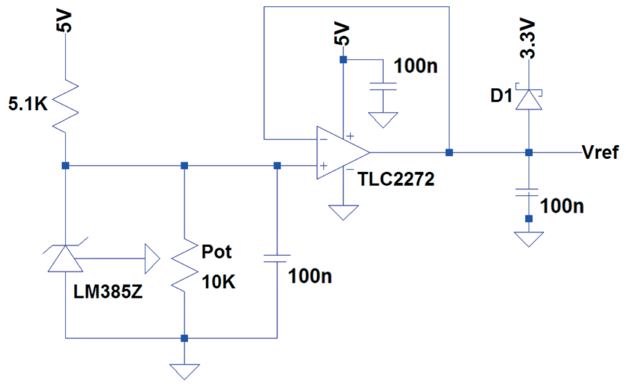
In this shield three different circuits have been implemented: one to adapt the DA output signal (control signal), another two to adapt the two measurement signals (potentiometer and tachometric dynamo signals) and, finally, one to create the offset for these signals. Figure 4 shows the circuit of those three signals.



(a) Signal conditioning circuit for the input of the A/D converter.



(b) Signal conditioning circuit for the output of the D/A converter.



(c) Circuit to obtain the desired offset.

Fig. 4: Three main circuits of the signal adapter

The signals must have change of gain and a change of offset both in the measurement channels (A/D conversion) and in the control signal channel (D/A conversion). These two actions are carried out separately (in cascade) to reduce the coupling between them and facilitate the analysis by the students.

The conditioning circuits for the input of the A/D converter and the output of the D/A converter are practically equal but changing the order of the actions. In the case of the conditioning chain of the A/D converter the order is: first the gain change and after that the offset change. For the D/A converter conditioning chain in the first stage the offset of the signal is changed and, then, it is scaled. In both cases, a low-

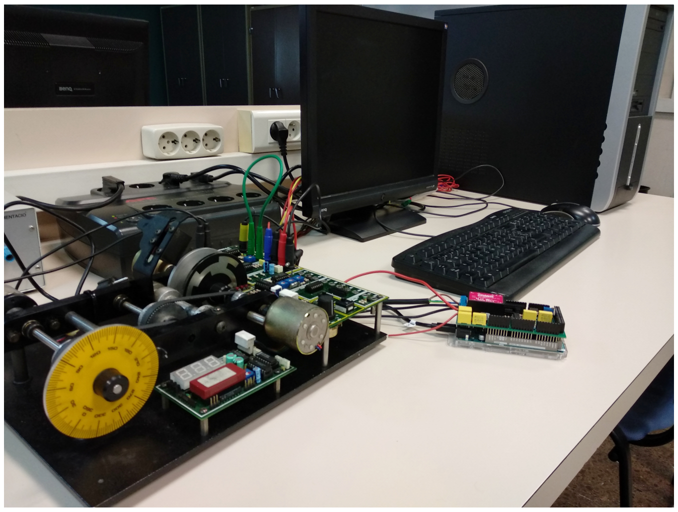
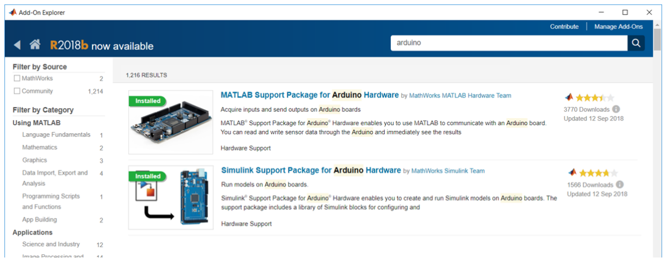


Fig. 5: Automatic control laboratory workplace.


Add-On Explorer

Contribute | Manage Add-Ons

R2018b now available

arduino

Filter by Source
MathWorks 2
Community 1,214

Filter by Category
Using MATLAB
Language Fundamentals 1
Mathematics 2
Graphics 3
Data Import, Export and Analysis 4
Programming Scripts and Functions 1
App Building 2
Applications
Science and Industry 12

1,216 RESULTS

Installed
MATLAB Support Package for Arduino Hardware by MathWorks MATLAB Hardware Team
Acquire inputs and send outputs on Arduino boards
MATLAB® Support Package for Arduino® Hardware enables you to use MATLAB to communicate with an Arduino board. You can read and write sensor data through the Arduino and immediately see the results
Hardware Support
3770 Downloads
Updated 12 Sep 2018

Installed
Simulink Support Package for Arduino Hardware by MathWorks Simulink Team
Run models on Arduino boards.
Simulink® Support Package for Arduino® Hardware enables you to create and run Simulink models on Arduino boards. The support package includes a library of Simulink blocks for configuring and
Hardware Support
1566 Downloads
Updated 12 Sep 2018

Fig. 6: Arduino MATLAB/Simulink Support Package.

pass filter with a low time constant has been designed in the last stage of the circuit in order to filter the possible peaks of the signal and the high frequency noise. The necessary reference signal has been created using a potentiometer and an LM385Z voltage reference whose output is connected to a voltage follower to minimize the effects of the load.

These devices have been implemented and an image of the final result in one of the lab workplaces can be seen on Figure 5. More information about the steps followed to implement the signal adapter can be found in three langues (English, Spanish an Catalan) on the repository linked in Duino-Based Learning (https://github.com/DuinoBasedLearning/Lab).

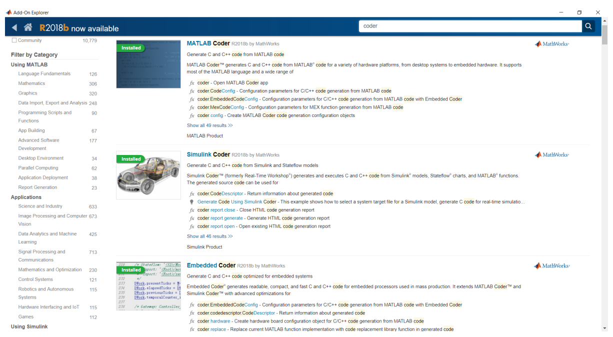
C. Software issues

This hardware change leads to modify the MatLab/Simulink based software used until now to install some toolboxes of the standard MatLab distribution to enable the data transfer from the computer to the plant and vice versa. The installed packages are: `MatLab and Simulink Support Packages for Arduino Hardware` in order to acquire inputs and send outputs for Arduino Boards, see Figure 6, and `MatLab Coder, Simulink Coder and Embedded Coder` for the generation of C and C++ code for embedded systems. This packages can be easily found in MatLab's tab home, in adds-on section, see Figure 7.

Once the support packages for Arduino have been installed, we have to differentiate and explain the two main methods that Simulink has for working with the Arduino board and the PC: `Normal Mode with Simulink IO` and `External Mode`. Both methods have their pros and cons, and, depending on the project, one method may prove to be more beneficial than the other.

- **Normal Mode with Simulink IO:** In this mode, the Simulink model, in our case containing the controller, runs in normal mode simulation on the PC and it communicates with an IO server code that runs on the Arduino board. This code manages the peripherals (AD and DA converters) of the microcontroller on the Arduino board and communicates with the simulation model (controller) on the PC. Thus, in this work mode, no code is generated corresponding to the controller to execute it in the microcontroller. The main disadvantage with this mode of operation is that the whole system does not work in strict real time as the controller code is running in the PC. This circumstance also affects the possible recording of signals entering and leaving the plant. That is, it is not possible to do a strict timing analysis of real-time data.
- **External Mode:** In this case, code is generated for the microcontroller both for peripheral management and for the control algorithm. This code is executed in real time in the microcontroller closing the control loop. On the other hand, the code generated and deployed in the microcontroller includes the management of communications with the PC and, in this way, a certain number of signals can be displayed in the Scopes of the Simulink model. This approach allows to obtain real-time data from the control loop and perform a precise time analysis of the signals since all the execution is done in the microcontroller in real time and the communication channel between controller board and PC is only used to transfer information. It is also worth to mention that, in External Mode, it is possible to modify the parameters of the controller without having to re-compile or re-download the model each time a parameter is changed. However, as a drawback and as it will be commented below, the limited bandwidth of the communication channel will bound from below the sampling period of the signals to be displayed on the Simulink Scopes.

Besides these two operation modes, it is worth to comment


Add-On Explorer

R2018b now available

coder

Community 10,779

Filter by Category
Using MATLAB
Language Fundamentals 126
Mathematics 306
Graphics 320
Data Import, Export and Analysis 248
Programming Scripts and Functions 90
App Building 67
Advanced Software Development 177
Desktop Environment 34
Parallel Computing 62
Application Deployment 38
Report Generation 23
Applications
Science and Industry 633
Image Processing and Computer Vision 673
Data Analytics and Machine Learning 425
Signal Processing and Communications 713
Mathematics and Optimization 230
Control Systems 121
Robotics and Autonomous Systems 115
Hardware Interfacing and IoT 115
Games 112
Using Simulink

Installed
MATLAB Coder R2018b by MathWorks
Generate C and C++ code from MATLAB code
MATLAB Coder™ generates C and C++ code from MATLAB® code for a variety of hardware platforms, from desktop systems to embedded hardware. It supports most of the MATLAB language and a wide range of
coder - Open MATLAB Coder app
coder.CodeConfig - Configuration parameters for C/C++ code generation from MATLAB code
coder.EmbeddedCodeConfig - Configuration parameters for C/C++ code generation from MATLAB code with Embedded Coder
coder.MexCodeConfig - Configuration parameters for MEX function generation from MATLAB code
coder.config - Create MATLAB Coder code generation configuration objects
Show all 49 results >>
MATLAB Product

Installed
Simulink Coder R2018b by MathWorks
Generate C and C++ code from Simulink and Stateflow models
Simulink Coder™ (formerly Real-Time Workshop®) generates and executes C and C++ code from Simulink® models, Stateflow® charts, and MATLAB® functions. The generated source code can be used for
coder.CodeDescriptor - Return information about generated code
Generate Code Using Simulink Coder - This example shows how to select a system target file for a Simulink model, generate C code for real-time simulatio...
coder.report.close - Close HTML code generation report
coder.report.generate - Generate HTML code generation report
coder.report.open - Open existing HTML code generation report
Show all 46 results >>
Simulink Product

Installed
Embedded Coder R2018b by MathWorks
Generate C and C++ code optimized for embedded systems
Embedded Coder® generates readable, compact, and fast C and C++ code for embedded processors used in mass production. It extends MATLAB Coder™ and Simulink Coder™ with advanced optimizations for
coder.EmbeddedCodeConfig - Configuration parameters for C/C++ code generation from MATLAB code with Embedded Coder
coder.codedescriptor.CodeDescriptor - Return information about generated code
coder.hardware - Create hardware board configuration object for C/C++ code generation from MATLAB code
coder.replace - Replace current MATLAB function implementation with code replacement library function in generated code

Fig. 7: MATLAB Coder, Simulink Coder and Embedded Coder packages.

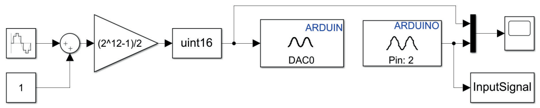


Fig. 8: Sampling time variability analysis.

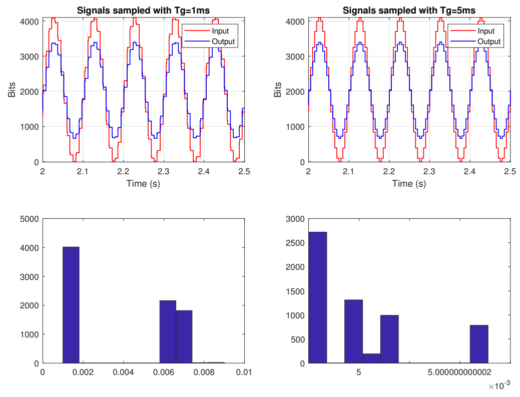


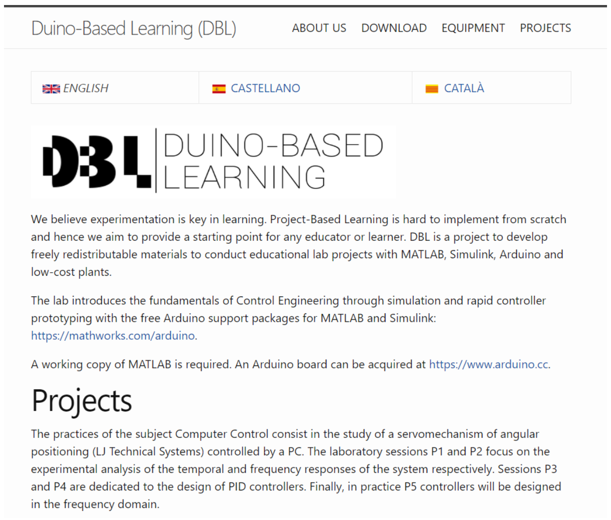
Fig. 9: Results with different sampling times for plotting in the Scopes. On the left, $T_g = 0.001s$, on the right $T_g = 0.005s$.

that there is another one called `Deploy to Hardware`. In this case, the generated code runs in the microcontroller as a standalone program without communicating with the PC and, so, it is independent of Simulink.

As it was mentioned above, a problem with `External Mode` operation is the limited bandwidth of the communication channel between the microcontroller board and the PC in which various relevant signals from the control loop are displayed graphically in Scopes. Therefore, it is important to find the minimum value of the sampling period (associated with the plotting) without loss of information. In order to do that, the Simulink model shown in figure 8 has been created. In it, a sinusoidal signal created in Simulink is compared with the same signal after passing it through the DA channel and getting back through an AD input channel.

This model has been run with different sampling times and the signals and their sampling time variability have been analyzed for each iteration. We have selected a sinusoidal signal of 10 Hz and started from a sampling time of 0.001 s verifying if there is loss of samples in the Scopes. If there are losses, the sampling period is increased until there is no loss of information. Figure 9 shows the result of two experiments done when calculating the loss of samples.

As we can see, from sampling periods higher than 5 ms, no data is lost. From this value on, the acquired signal period variability is very small as it can be appreciated on the histogram because all the samples occur at 5ms of sampling time with a very small jitter. In contrast, when a sampling time lower than 5 ms is selected, for example in Figure 9, 1 ms, half of the samples are around 1ms but the other half are between 5 and 6 ms.



We believe experimentation is key in learning. Project-Based Learning is hard to implement from scratch and hence we aim to provide a starting point for any educator or learner. DBL is a project to develop freely redistributable materials to conduct educational lab projects with MATLAB, Simulink, Arduino and low-cost plants.

The lab introduces the fundamentals of Control Engineering through simulation and rapid controller prototyping with the free Arduino support packages for MATLAB and Simulink: https://mathworks.com/arduino.

A working copy of MATLAB is required. An Arduino board can be acquired at https://www.arduino.cc.

Projects

The practices of the subject Computer Control consist in the study of a servomechanism of angular positioning (LJ Technical Systems) controlled by a PC. The laboratory sessions P1 and P2 focus on the experimental analysis of the temporal and frequency responses of the system respectively. Sessions P3 and P4 are dedicated to the design of PID controllers. Finally, in practice P5 controllers will be designed in the frequency domain.

Fig. 10: Homepage of Duino-Based Learning.

III. Duino-Based Learning (DBL)

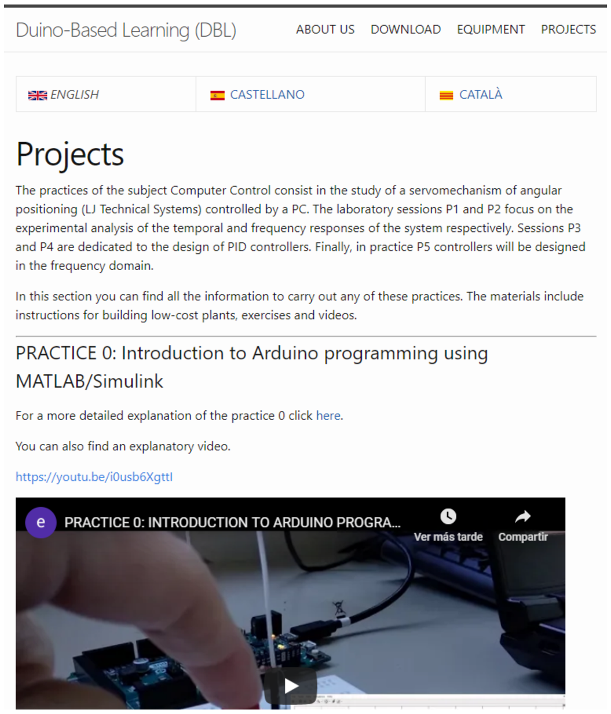
A. About DBL

As it is known, projects are hard to implement from scratch but it is the best way to put into practice and assimilate all the techniques and concepts of the theoretical sessions and, in addition, to realize the differences between the real implementations and their theoretical description. In this sense, Duino-Based Learning (DBL) provides a set of projects as a starting point for any educator or learner. In it you can find build instructions for all setups, MATLAB live scripts with exercises, Simulink models and walk-through videos. All this material is available in three languages English, Spanish and Catalan. The web page can be found in https://duinobasedlearning.github.io/. Figure 10 shows the layout of the platform.

As it can be seen, the web page is divided in five sections. The first one is the homepage, where all the main information is embedded; the second one, `about us`, shows the authors with a brief description of their activity. In the third one, `download`, you can download all the files available in the repository. In the fourth one, `equipment`, it is explained all the instrumentation used to carry out the projects and how they are interconnected. Finally, in the last section, `projects`, the user can find the five projects that have been prepared with instructions for building low-cost plants, exercises and videos, see Figure 11. In every lesson, the first link corresponds to MatLab scripts with data, graphs and explanations, while the second one corresponds to the tutorial videos where theoretical issues are explained and the real behaviour of the plant can be seen.

B. Projects

Along all the web page there are five different practices to work on that include some proposed exercises. The contents range from a very basic introduction to Arduino programming using MatLab/Simulink to the design of controllers in the frequency domain.


Duino-Based Learning (DBL)

ABOUT US DOWNLOAD EQUIPMENT PROJECTS

ENGLISH | CASTELLANO | CATALÀ

Projects

The practices of the subject Computer Control consist in the study of a servomechanism of angular positioning (LJ Technical Systems) controlled by a PC. The laboratory sessions P1 and P2 focus on the experimental analysis of the temporal and frequency responses of the system respectively. Sessions P3 and P4 are dedicated to the design of PID controllers. Finally, in practice P5 controllers will be designed in the frequency domain.

In this section you can find all the information to carry out any of these practices. The materials include instructions for building low-cost plants, exercises and videos.

PRACTICE 0: Introduction to Arduino programming using MATLAB/Simulink

For a more detailed explanation of the practice 0 click here.

You can also find an explanatory video.

https://youtu.be/i0usb6XgttI

Fig. 11: Projects section of Duino-Based Learning web page.

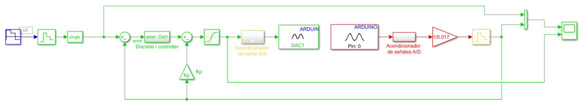


Fig. 12: Speed Control IP controller: implementation.

The list of practical sessions (projects) with a brief description of their content is as follows.

Practice 0: Introduction to Arduino programming using MatLab/Simulink.

1) Digital output.
2) Digital input.
3) Analog output.
4) Analog input.

Practice 1: Analysis of the time response of a digital control system.

1) Speed control.
   - Open-loop response.
   - Closed-loop response.
   - Experimental evaluation of closed-loop performance.
2) Position control.
   - Open-loop response.
   - Closed-loop response.
   - Experimental evaluation of closed-loop performance.

Practice 2: Analysis of the frequency response of a digital control system.

1) Nyquist diagram calculation.

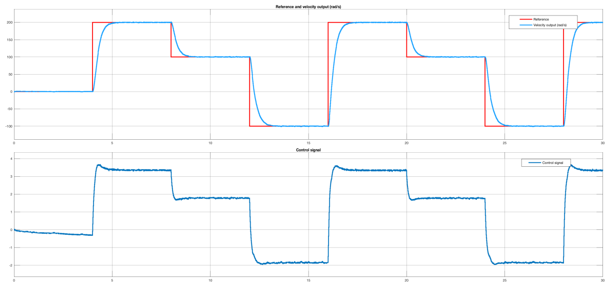
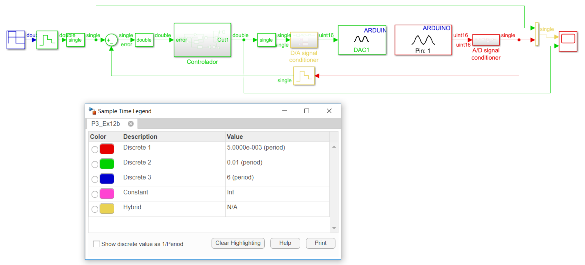


Fig. 13: Speed Control IP controller: experimental results.



Sample Time Legend

P3_Ex12b

Color	Description	Value
🟥	Discrete 1	5.0000e-003 (period)
🟩	Discrete 2	0.01 (period)
🟦	Discrete 3	6 (period)
🟪	Constant	Inf
🟨	Hybrid	N/A

Show discrete value as 1/Period | Clear Highlighting | Help | Print

Fig. 14: Position PID controller implementaion.

2) Experimental Nyquist diagram of the discrete-time system.
3) Application of the Nyquist criterion.
4) Obtaining the theoretical Bode diagram.

Practice 3: Design and implementation of PID controllers.

1) Design, by pole placement, and implementation of a PI controller for speed output.
2) Design, by pole placement, and implementation of a PID controller for position output.

Practice 4: Improvement of PID controllers implemented in Practice 3.

1) Design, by pole placement, and implementation of an I-P controller for speed output.
2) Design, by pole placement, and implementation of an I-PD controller for position output.

Practice 5: Design of controllers in the frequency domain.

1) Determination of the gain and phase margins for the system.
2) Design of a lead controller.
3) Analysis in simulation.
4) Implementation of the lead controller.

C. Repository

All the projects explained in the web page and the Gerber files used for the shield implementation are freely available in GitHub. The objective of this project, together with helping learners to understand the basics of digital control and educators to have materials on which to base their lessons or generate new ones, is to encourage people, in general, to contribute with new ideas or better implementations and, thus, obtain a continuous update of the web page. The repository can be found in https://github.com/DuinoBasedLearning/Lab.

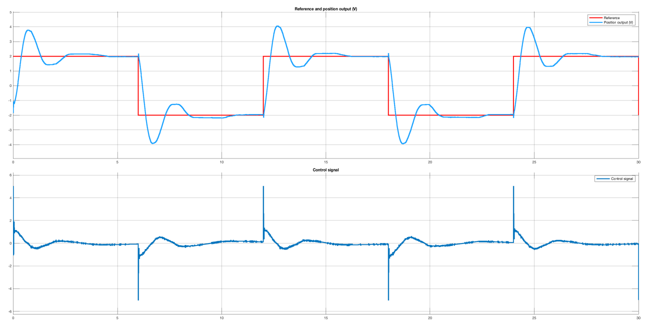


Fig. 15: Position PID controller experimental results.

IV. An example of a practice

PID controllers [15], [16] are one of the most popular controllers used in industry. Due to this, PID controllers are part of the experimental works students perform during their laboratories.

As an example, two of the didactic experiences carried out by the students are described below.

The Figure 12 shows the implementation of an IP controller. This controller will be used to control the speed of the DC motor. The IP control, unlike the PI controller, incorporates the integral action in the direct chain and the proportional action in the feedback chain. This means that the loop system does not have an additional zero in the closed-loop transfer function. For this reason, the response to the step usually presents an overshoot lower than what a PI with the same closed-loop poles would obtain. As you can see the implementation of the controller in MATLAB/Simulink is very easy and maintains the structure of the controller, which facilitates its comprehension to students.

Figure 13 shows the closed-loop time response, following the indicated instructions. As it can be seen the step response is fast and without overshoot. One of the characteristics of IP controllers.

Figure 14 shows the implementation of a complete PID controller. In addition to the controller implementation, using a color code, the different sampling periods used are shown. Some just to display purposes, others to close the loop.

Figure 15 shows the time response of the closed-loop system in position control and using the PID control shown above. As it can be seen, null steady-state error is obtained in tracking step references with a slight overshoot. The response presents the imperfections typical of the non-linearities existing in the real process.

During the practice sessions, the controllers are tuned by different methods, being the pole placement one of them [14], [17], [18].

V. Conclusions

This work has presented the contents and teaching material used in the practice sessions of a discrete-time control course. All material is available on the Internet and it can be freely used. All the work is based on the use of interactive control design tools such as MATLAB and Simulink to help students understand what is a sampled-data system and how to deal with its control. The proposed experiments are based on low cost hardware, i.e. Arduino Due boards, that are programmed using the automatic code generation tools that are included in MATLAB/Simulink. This approach can encourage people to get into the digital control field since the software environment greatly facilitates tasks related to programming embedded systems.

Acknowledgments

This work has been partially funded by the Spanish national projects MICAPEM (ref. DPI2015-69286-C3-2-R, MINECO/FEDER), DPI2017-85404-P and the donation of Mathworks UPC-I-01523 .

References

[1] J. C. Plaza, A. M. Florido, E. P. Garca, F. R. Montero, and R. C. Palomino, "Entorno docente universitario para la programacin de los robots," *Revista Iberoamericana de Automtica e Informtica industrial*, vol. 15, no. 4, pp. 404–415, 2018. [Online]. Available: https://polipapers.upv.es/index.php/RIAI/article/view/8962

[2] R. Costa-Castelló, V. Puig, and J. Blesa, "On teaching model-based fault diagnosis in engineering curricula [lecture notes]," *IEEE Control Systems*, vol. 36, no. 1, pp. 53–62, Feb 2016.

[3] J. M. Diaz, R. Costa-Castelló, R. Muoz, and S. Dormido, "An interactive and comprehensive software tool to promote active learning in the loop shaping control system design," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2017.

[4] R. Costa-Castelló, N. Carrero, S. Dormido, and E. Fossas, "Teaching, analyzing, designing and interactively simulating of sliding mode control," *IEEE Access*, vol. 6, no. 1, pp. 16 783–16 794, December 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2815043

[5] F. Esquembre, "Easy java simulations: a software tool to create scientific simulations in java," *Comput. Phys. Commun.*, vol. 156, no. 2, pp. 199–204, January 2004.

[6] J. Chacn, M. Guinaldo, J. Snchez, and S. Dormido, "A new generation of online laboratories for teaching automatic control," *IFAC-PapersOnLine*, vol. 48, no. 29, pp. 140 – 145, 2015, iFAC Workshop on Internet Based Control Education IBCE15.

[7] J. Sobota, R. PiSl, P. Balda, and M. Schlegel, "Raspberry pi and arduino boards in control education," *IFAC Proceedings Volumes*, vol. 46, no. 17, pp. 7 – 12, 2013.

[8] P. Reguera, D. Garca, M. Domnguez, M. Prada, and S. Alonso, "A low-cost open source hardware in control education. case study: Arduino-feedback ms-150," *IFAC-PapersOnLine*, vol. 48, no. 29, pp. 117 – 122, 2015.

[9] F. Candelas, G. García, S. Puente, J. Pomares, C. Jara, J. Pérez, D. Mira, and F. Torres, "Experiences on using arduino for laboratory experiments of automatic control and robotics," *IFAC-PapersOnLine*, vol. 48, no. 29, pp. 105 – 110, 2015.

[10] R. Barber, M. Horra, and J. Crespo, "Practices using simulink with arduino as low cost hardware," *IFAC Proceedings Volumes*, vol. 46, no. 17, pp. 250 – 255, 2013.

[11] M. ISHIKAWA and I. MARUTA, "Rapid prototyping for control education using arduino and open-source technologies," *IFAC Proceedings Volumes*, vol. 42, no. 24, pp. 317 – 321, 2010.

[12] S. Dormido, "Control learning: Present and future," *Annual Reviews in Control*, vol. 28, no. 1, pp. 115–136, 2004.

[13] B. Kuo, *Digital Control Systems*. Oxford University Press, 1995.

[14] R. Costa Castelló and E. Fossas, *Sistemes de Control en Temps Discret*. Edicions UPC, 2014, iSBN: 978-84-9880-492-8.

[15] K. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*. Dover Pubs., 2011.

[16] K. Åström and T. Hägglund, *Advanced PID Control*. ISA-The Instrumentation, Systems, and Automation Society, 2006. [Online]. Available: https://books.google.es/books?id=XcseAQAAIAAJ

[17] R. Longchamp, *Comande Numériques de Systeèmes Dynamiques. Cours d'Automatique*. Lausanne: Presses Polytechniques et Universitaires Romandes, 2006.

[18] K. Ogata, *Discrete-time Control Systems*. Prentice Hall, 1994.