TEACHING EMBEDDED/IOT TO ALL ENGINEERS

Paulo Ferreira[1], Benedita Malheiro[2], Manuel Silva[2], Pedro Guedes[2], Jorge Justo[1], Cristina Ribeiro[3], Abel Duarte[1]

[1]Polytechnic Institute of Porto - Engineering School (PORTUGAL)
[2]INESC TEC / Polytechnic Institute of Porto - Engineering School (PORTUGAL)
[3]I3S-INEB / Polytechnic Institute of Porto - Engineering School (PORTUGAL)

Abstract

In a traditional Engineering curriculum, computers replaced electronic calculators that replaced slide rules, always with the purpose of calculating more, better, and faster. Nowadays, besides being interconnected, computers are embedded in many devices, from smartcards to automobiles, with diverse functionalities and executing a wide range of tasks. Modern engineers use programming languages (with computers) not only for calculations, but also for automating software or equipment. While offering multiple possibilities, embedded processors place also serious restrictions on the programmer, as some tools have serious limitations and deviations from "normal programming", for instance while debugging. The chip shortage due to the COVID-19 pandemic has further complicated the situation, as many development boards are unavailable, have astronomical prices or both.

This paper discusses the use of Python and Micropython for teaching programming, in the context of a Project Based Learning experience involving a multicultural and diverse team. The use of Python allows the use of an Open-Source language with a wide variety of applications like scientific computing, data mining, web engineering, system management and many others. MicroPython allows the reuse of the same language and associated programming knowledge in small embedded platforms. The interactive nature of Python facilitates the debugging of the built systems, while the abstractions provided by the language ease the task of porting the software to a different development board. The use of a board simulator can mitigate the reduced availability of boards, due to chip shortage or any other procurement difficulty.

When compared with the traditional embedded choice of the C programming language, the use of MicroPython requires more memory, a greater processing power and is slower. These arguments may not be as sound as they seem, because: the price differential to a better processor may be negligible; the memory required to support MicroPython may be the same required to support the networking libraries needed; the time critical Python functions can be replaced by faster functions in C; MicroPython proves a faster development, so a faster time to market.

This approach has already been validated by our students, with good acceptance and notable results in several projects.

Keywords: Programming, Embedded Systems.

1 INTRODUCTION

The European Project Semester (EPS) [1,2] is a project-based, international teamwork initiative, offered by a network of 19 Higher Education Institutions (HEI) located in 12 different European countries, that replaces the design capstone semester of undergraduate engineering degrees. The network develops transnational projects together and exchanges students, staff, ideas, and best practices, expanding the influence and the pedagogical experience of the providers.

At the *Instituto Superior de Engenharia do Porto* (the Polytechnic of Porto Engineering School), the EPS programme (EPS@ISEP) is active only on the spring semester. It provides 30 European Credit Transfer System Units (ECTU), with 20 ECTU assigned to the project module and 10 ECTU equally divided by five support modules: Energy and Sustainable Development, Ethics and Deontology, Foreign Language and Culture, Marketing and Communication, and Project Management and Teamwork. During its 12-year existence most students were international although the programme is open to international and local students. To develop the project, teams of 4 to 6 students are created, aiming to maximize the group's diversity and multiculturalism. Each group has the freedom to choose an open-ended project from a wide list previously created by the supervisors, considering feasibility, the programme's objectives, and the availability of the different resources.

An additional "Technological Crash-Course", about electricity, electronics, embedded systems, and programming has been added in the last years, as many of the projects involve automation and/or programming. Due to the diversity of the students' degrees, this short course acts as an introduction to the subjects to many students, while providing a refresher for some of them.

An introduction to programming is taught in most of the Engineering Degrees, using different languages, but always with the purpose of providing support for the other courses that compose the Degree. Considering that, to teach programming as a support module of a capstone project may seem redundant and unnecessary and needs to be justified. Besides providing students with strong notions of computational thinking, programming is useful to perform calculations, automate tasks and control all types of equipment. In the beginning of a degree, students are still not aware of the typical tasks they will need to perform professionally and do not possess the required mathematical foundations to support them.

Embedded systems are typically taught in electrical/electronic engineering degrees, using a digital systems perspective. The Hardware/Software Co-Design perspective, where one needs to choose between hardware or software implementations, is mainly taught in advanced degrees, while a typical embedded systems course consists in exploring the hardware and software design possibilities offered by a typical microcontroller, using C, C++ and/or Assembly programming languages. These approaches are consistent with the engineering objective of providing the maximum of functionality with the minimum of resources and demand obviously a strong hardware/software knowledge, usually focused on a single computing architecture (microcontroller family or single board computer).

As a contrast, on the EPS@ISEP programme the objective is not the building of an embedded system but to consider how the inclusion of an embedded system in the final product can improve the product, help to explore different functionalities and open new possibilities of integration in larger systems. So, the development platforms need to be chosen not to minimize the required computational power, but to maximize the flexibility of the development.

The COVID-19 pandemic has brought additional issues, with volatile prices and reduced availability of many electronic components, that complicate their purchase, especially when going through the bureaucratic red tape process enforced by many HEIs.

2 IMPLEMENTATION

The technological “crash-course” implemented in the ESP@ISEP programme, consists of 8 in-person classes of 2 hours each (a total of 16 hours), during the first 4 weeks of the course, supported by a Microsoft Teams discussion group and file repository for additional on-line student support. The timing of the course is a compromise that aims to provide the students with the required knowledge when they need it (the initial design phase), while giving them time to reflect and experiment during the course.

The course consists of a series of topics that are always being adapted and improved. The topics are usually:

- Open Source
- Electricity and Electronics
- Microcontrollers and Single Board Computers
- Python and MicroPython Programming

One of the requirements of the projects is that the final product should be Open Source, to facilitate its adoption, reproduction, and evaluation. As such, every team has a public wiki, where all the documentation related to the project is shared and stays archived after the projects have ended. Some notions about Open Source and the correct respect of all types of software licenses are given to the students.

The Electricity and Electronics topic provides some notions of Direct Current (DC) electricity, use of batteries, power supplies and breadboards, to support the elaboration of electronic prototypes. The use of the TinkerCad simulator allows the realization of virtual experiments without the danger damaging real components. The manipulation of real components causes some initial fear on the students, that can be avoided using a simulator to introduce electronic circuits. After some experience with virtual electronic circuits, the students will be more comfortable to manipulate real circuits.

In the original EPS@ISEP edition (2010/2011) the groups used Arduino Uno boards [3,4] to automate their projects, as they fitted the requirements of the project and were easy to use. During the EPS@ISEP editions, the Arduino platform was used (and reused) many times due to its low-cost and ease of use.

Sometimes, due to specific requirements, other solutions like the Arduino Mega, Arduino Micro, Raspberry Pi, TI TivaC Launchpad, Bluno Beetle, custom PIC16 boards, ESP8266 or ESP32 boards were used.

The Arduino software used in many ESP@ISEP projects, includes an Integrated Development Environment (IDE) that eases simple embedded programming tasks, and the software is supported by online simulators such as Tinkercad (https://www.tinkercad.com). But the quality of the libraries is very uneven, many compiler warnings are hidden by default, and debugging is hard to do.

On the last editions of EPS@ISEP, the use of the Python programming language was tested, focused on the MicroPython implementation [5,6]. This has resulted from a long analysis of the problem and of a careful evaluation of all the alternatives.

The Python language can be used in many platforms, from Android to Windows, and for many purposes from scientific calculations to process automation. To introduce the students to Python, one can use an online interactive course implemented using a series of Jupyter Notebooks. The Jupyter hub installation only uses a low-cost virtual server and gives the students the possibility of learning Python and experimenting with code fragments, without the need to install additional software on their computers. As all the students code is placed on the Jupyter Hub, the teachers have full access to the students' source code for helping them on the learning (and debugging) process. As an alternative to installing a Jupyter Hub instance one can use Amazon SageMaker or Google Colab as they are modified Jupyter Hub installations.

The use of MicroPython demands a more powerful processor than the AVR328 8-bit processor typically used on Arduinos. MicroPython requires the use of a 32-bit processor such as STM32F3, ESP8266, ESP32 or RP2040 and with greater RAM and flash ROM capacities. This could apparently lead to a serious price increase but sometimes leads to cheaper prices. As an example, the price of a RP2040 based board (Raspberry Pi Pico – 4.60 Euro in May 2022) is normally lower than the price an Arduino Uno. The use of a different and more performant processor is also required if one to desires a WiFi or Bluetooth connection, so a change to different processor may be needed independently of the use of MicroPython.

The permanent changes in the availability and prices of boards due to the COVID-19 semiconductor shortage are an additional reason to use MicroPython, as MicroPython eases the change to a different board in the middle of the development. The debugging is easier than with Arduino as one can execute MicroPython interactively on the target platform.

It is obvious that the performance of MicroPython is not comparable to the performance of C or C++. But the objectives of the EPS@ISEP are the construction of a prototype or proof of concept (PoC), where the performance many times is not the most pressing constraint. In the rare cases where more performance is needed the MicroPython code can be "pre-compiled" into ROM or an additional C library can be built into MicroPython, with the original MicroPython source code serving as an already pre-debugged template.

Writing the desired code twice, in two different languages may be faster that writing it only once in C/C++. This is because initially one is debugging not only our source code, but also our hardware that may not behave as predicted/specified/desired. The interactive nature of MicroPython lets us avoid the compile-download delay, providing a richer and faster debug experience without the need of expensive hardware debugging aides (as emulators or JTAG probes).

The use of the Wokwi online simulator (https://wokwi.com) supports hardware/software development, as it simulates the use of MicroPython on ESP32 and Raspberry Pico boards, providing simulated circuits and network connections that are used to implement/use Internet of Things (IoT) services

3 RESULTS AND CONCLUSIONS

On the current EPS@ISEP edition (2021/2022) a total of 24 students from 8 different countries are currently enrolled. As many COVID-19 were already lifted, the classes were fully in-person and Microsoft Teams was used as a support platform. The student adhesion was very good considering their choice of development technologies that can be seen on the support wikis, and from the persistent student's use of the Jupyter Notebooks, after the end of the technological crash course.

The order of the different topics in the current edition was:

- Open Source

- Python Programming
- Electricity and Electronics
- Microcontrollers and Single Board Computers
- MicroPython use and Programming

The short Open Source topic provides a general introduction. The Python programming topic is placed in the beginning to provide students with a longer and softer introduction to programming, and as the Python support materials can be used whenever and wherever the students desire, this increases the effectiveness of the course.

At the time of writing (May 2022) the current EPS@ISEP edition has not ended but the feedback is positive, both from the students and the project supervisors, and the technological “crash-course” will be adapted and developed for future EPS@ISEP editions. The use of MicroPython (Python) allows an interactive and attractive introduction both to programming and embedded systems, that can be used in many different engineering tasks, while support a wide range of embedded boards. The use of web-based simulators (TinkerCad and Wokwi) and Open-Source software lowers the course’s cost and allows an easier replication of the final projects.

This strategy aims to teach embedded programming to a wide variety of engineers, leading to a greater “smartification” and IoT enabling of many engineering projects, always in a sustainability oriented context, as can be seen from the EPS@ISEP reports [7].

REFERENCES

[1] M. F. Silva, B. Malheiro, P. B. Guedes, P. D. Ferreira, and A. Duarte, ‘The European Project Semester at ISEP (EPS@ISEP) programme: Implementation results and ideas for improvement’, in *Proceedings of the 45th SEFI Annual Conference 2017 - Education Excellence for Sustainability, SEFI 2017*, Azores, Portugal, 2017, pp. 129–130.

[2] A. J. Duarte, Duarte, B. Malheiro, E. Arnó, I. Perat, M. F. Silva, P. Fuentes-Durá, P. Guedes, P. Ferreira, 'Engineering Education for Sustainable Development: The European Project Semester Approach,' in IEEE Transactions on Education, vol. 63, no. 2, pp. 108-117, May 2020, doi: 10.1109/TE.2019.2926944.

[3] M. Margolis, *Arduino Cookbook*, 2nd ed. Sebastopol, CA: Oreilly & Associates Inc, 2012.

[4] D. Wheat, *Arduino Internals*. New York: Apress, 2011.

[5] C. Bell, *MicroPython for the Internet of Things*. New York, NY: Apress, 2017.

[6] N. H. Tollervey, *Programming with MicroPython*. Sebastopol, CA: O’Reilly Media, Inc., 2017.

[7] M. F. Silva, A. J. Duarte, P. D. Ferreira, P. B. Guedes, 'Robotics and the European Project Semester' In B. Malheiro, P. Fuentes-Durá (Ed.), *Handbook of Research on Improving Engineering Education with the European Project Semester* (pp. 205-219). IGI Global 2022. https://doi.org/10.4018/978-1-6684-2309-7.ch011.