First-year integrative project for computer and software engineering students at Polytechnique Montréal

Jérôme Collin
Polytechnique Montréal
Department of Computer and Software Engineering
jerome.collin@polymtl.ca

Abstract *– This paper presents the most important aspects of the first-year project for students in computer and software engineering at Polytechnique Montréal. A small robot is used to introduce students to hardware and software concepts of a complete and autonomous small computer. A custom robot was designed for this course and this gives the flexibility to introduce some concepts in a specific way. The fact that both software and hardware are considered makes it particularly challenging for the students but also for the teaching team. The robot itself is described but also the project structure and how students can progress in this context. The evaluation and the support offered to the students are also explained.*

Keywords: first year, computer, software, integrative project, course structure, robot.

1. INTRODUCTION

The first-year project in computer and software engineering at Polytechnique Montréal has been a great success and students really appreciate it. Therefore, this paper describes how it has been developed and the context surrounding it. It underlines how details are important to provide a good learning environment to the students.

In the present case, the biggest problem was the fact that students must start a project with very limited technical background. Another problem was to propose a project course which would take key concepts from various other courses in the curriculum during this same first year and include them in the project. Consequently, this integrative approach would require students to pass valuable time in a laboratory where they would experiment with knowledge they have learned or are learning while the project progresses.

During this first year, students follow two courses on hardware, the first on basic digital design and the second on computer architecture. Two courses on programming are also scheduled, the first on procedural programming and the second on object-oriented programming (both with C++). Two other courses are also proposed on methodology. The first one is a general introduction to computer engineering and the second is specifically on fundamental software engineering. Some mathematical courses complete the first-year program.

Therefore, we were looking for a project in which students could program a hardware device they have to understand. They would work in teams and they would follow a specific methodology. This course would be placed during the second semester of the first year. Beyond this, very little was assumed. However, we didn't want to make some kind of "big homework" but a complete project where, at the end, students would have a complete hardware system, with all its programming aspects and complex input/output relationship. Therefore, at the end of their first year, they would be in a position to understand a little computer system. We didn't go as far as the "From NAND to Tetris" course [9] but we certainly share some objectives with this approach.

Using robotics to teach programming concepts is a method that has gained momentum [6] and can be quantified [4,7]. It was decided that our programming base would be a small mobile robot. This robot would be designed specifically for this project course as opposed to the E-Puck robot which is designed to be used in a large spectrum of teaching activities [8]. The robot would have to be easily assembled by students (including soldering of electronic parts) and would move around trying to interact with other robots and/or objects in its environment at the end of the semester. The capacity to install many different sensors and to use the robot in various situations was also important. Approaches that use Lego Mindstorms [5] or some kind of robot kit were excluded as a robot base because we wanted our students to grasp the hardware and understand the fundamentals without usage of high-level libraries or preprogrammed pieces of code.

2. THE ROBOT

Humans are fascinated by moving objects. It is something we can appreciate with our eyes naturally. Programming computer is seen as hard and not intuitive. Movement in a system can increase motivation and understanding of programming [1]. Having a robot would give us an interesting base to teach programming, digital design and computer architecture. Something as complex as a robot, even a very simple one, would be challenging enough, to the point that methodology and team work would be necessary and applied in an interesting context. Thus, a small robot became at the heart of our first-year project.

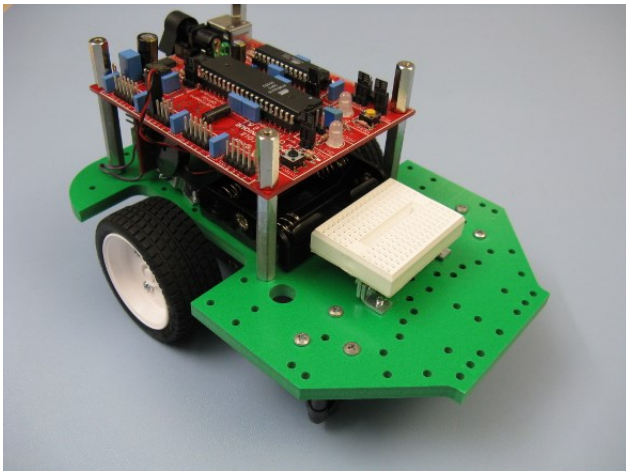

Figure 1. The robot

Great care was taken in the design of the robot. The goal was to come up with a mobile that could be easily assembled and repaired when necessary. Standard and highly available electronic parts would have to be soldered. A low cost was also a priority. It was also considered important to easily connect the robot to a PC to quickly download programs to the microcontroller. Modularity was desirable. The main sections of the robots are as independent as possible one from another. We ended up with a robot design (figure 1) that fulfilled all our criteria after a few iterations.

The base of the robot is made of simple PVC plastic to which various other main components are fastened. Some off-the-shelf wheels and electric motors take place underneath this plastic base. A classic H-Bridge electric motor drive is located at the rear. Towards the front, many holes in the PVC base are used to install various sensors. On top, a custom motherboard is the most important part. Two Atmel AVR microcontrollers are at the center of the system. The first is an ATmega324PA for general purpose usage. The second is an ATmega8 and is used to program the first one easily and it's also, with its special firmware, a USB peripheral. Thus, this motherboard can be easily programmed and powered by a USB cable. This architecture is close to the Arduino architecture but without the programming framework. The Arduino framework is excellent for the beginners but it hides all the lower level details that are exactly the ones we want our students to understand.

Right from the beginning, we wanted the robot design to be completely open source. This means that we publish on the course web site [2] what it takes to produce a robot. We even give the instructions to install the compiler on the PC for students who begin with Linux. This makes it possible to program the robot on a laptop or on a desktop at home without proprietary software licenses. The robot costs about $200.00 Canadian dollars. It comes in a plastic box and it has to be assembled. Over 500 have been produced since 2006 and more than 1000 students have assembled one by team of two during a project course.

3. HARWARE AND SOFTWARE

The first thing students will have to do is to assemble to robot, especially to solder the electronic components. This is a manual activity, obviously. Some have exposed the values of manual work [3]. We consider that first-year students benefit from this experience and they enjoy it.

However, it is also true that students don't design real hardware because we give them a robot that is already designed. With the limited background they have, and also because they are not studying in electrical engineering, we consider that this is acceptable. On the other end, this is a great opportunity to study an existing design and learn from it. In any case, the main hardware concepts of the course will be found inside the main microcontroller. This is where we find what we want the students to learn: a CPU, timers, UARTs, interrupt controllers, I/O lines, memories, buses, registers, and so on. How all these digital components work is taught in another course that students follow during the same semester.

This project-based course remains focused on programming in C/C++. No assembly language is used on the robot. To make links with the hardware structure, students start by writing short programs. Most of them will make use of precise and limited hardware resources at the beginning. Therefore, it is almost impossible for students to separate hardware and software when they write these programs. Doing so, they also have to mix concepts they have learned in other courses during this

first year. This is where this project becomes integrative in its nature. Understanding the dataflow between registers in the microcontroller and the interaction with variables and C procedures is critical. These concepts are the fundamental principles of any computer system. These programs are short but it takes time to write and to understand them.

To complete the software picture, we introduce Linux as a programming platform, including the basic shell commands. The emphasis is also placed on usage of a version control system, SVN. Moreover, we want students to be able to form a simple static library with a Makefile at some point in the project. This structure will introduce a basic software engineering methodology as well.

On the hardware side, we have to explain principles that students don't know about. PWM to drive DC electric motors with an H-Bridge is one of them. A basic review of electric circuit is also necessary when it is time to put electronic parts on a breadboard at the front of the robot. Obviously, how to solder and how to crimp connectors are also complementary notions that we give, sometimes using videos on the course web site.

4. THE COURSE, WEEK BY WEEK

Because this is a first-year project, students work by team of two at the beginning and they receive assignments on a weekly basis. The real final project will only begin around week number 10. This gives time to other courses of the same semester to progress and to introduce fundamental concepts that will be reused in the project course.

This also gives 8 weeks to a psychologist to teach interpersonal and team interaction skills. This part of the project introduces students to subjects like: leadership, teamwork dynamic, teamwork models and cohesion. How to organize efficient meetings and to define roles in a team are also discussed. The project will offer a real situation to students to demonstrate this know-how later in the project. The psychologist will continue to follow every team during the project after these 8 first weeks.

On the technical side, assembling the robot is the first assignment in this project. However, this usually doesn't take more than 7 to 10 days to finish. It is interesting to observe how much rhythm it gives to the course early in the semester. It can almost be viewed as a team building exercise as well because the degree of interaction is high. In a first-year project, students know very little each other sometimes and this helps even the communication between teams. Teaching assistants and staff members are also involved to help students and this increases interaction as well.

A team of two students can now begin the second assignment of the semester, the introduction to the motherboard. The first exercise will be to install a program that is on the course web site as an example and that is about only 10 lines of code. This program exposes how the input/output ports work. The first program we ask students to write will be to control a simple LED and the second one will be to read the output of a simple tact switch on the motherboard.

By the beginning of the third week of the course, we prepare students to write a software finite state machine (FSM), a concept they have learned a semester before but in the context of a digital design course. FSM are important to eventually develop a robot with automated behaviors.

The fourth week is when students start using the motherboard to control the wheels. Explanations about the H-Bridge circuit and Pulse Width Modulation (PWM) are also provided. Usually, some debugging with multi-meters and oscilloscopes are necessary and students are encouraged by the teaching assistants to use laboratory equipment.

The fifth week is an important turning point of the semester because students will be introduced to the internals of the main microcontroller and its architecture. It's time for the more complex notions: interrupts, timers, configuration registers, etc. These concepts are difficult to understand. That's why we propose to students to rewrite some short programs that have produced previously but instead of just using I/O ports to achieve the correct results, they have to use internal peripherals of the microcontroller. For example, using interrupts to read the output of a tact switch instead of using a polling method. Another example is how to use a timer to generate PWM to control the motor speed and to avoid a busy-wait programming structure to get the same results. This approach is interesting because it shows an important distinction between concepts even if the behavior of the microcontroller, from an external point of view, is the same. However, internally, the program and the electronic modules used are different. Students can also quickly reprogram their microcontroller with a program they wrote just a few weeks ago and realize that nothing has changed externally but the efficiency has increase when proper electronic modules are used.

Usually, by the sixth and seventh weeks of the semester, students are busy and the amount of work in other courses has grown. Some midterm exams are

scheduled in these courses and students feel they have to pass less time on the project. To give a chance to students, the pace is reduced in the project. Easier assignments are proposed during these two weeks. The overview of hardware resources continues with the UART (to communicate back some values to the PC), the Analogue to Digital Converter (ADC) and an external I^2C serial memory. Usually, these assignments are straightforward for most students, especially if they have understood correctly the details of the previous week.

Another turning point occurs again around week number eight. First, we regroup two teams of two to form one team of four. This will force students to develop code in a bigger group, something they have never done before. What we propose is also completely different. We want students to compare code they have written and that is currently in their SVN repositories. We want them to evaluate what is good and not so good. With what they want to keep, we ask them to form a static library of code that they can reuse for the remaining weeks of the semester. Obviously, this assignment forces them to talk and interact. This work is more software engineering oriented than what was covered during the past few weeks. This is also time to reorganize what was developed separately since the beginning.

During the ninth week, students have to write a much longer program. They reuse their library written the week before and they have to distribute between team members the responsibilities to develop pieces of code. This last assignment before the final project is an opportunity to watch teams that can do it with a relative easiness and the ones we will have to follow closely because they have problems to bring all their parts together. Usually, it is asked to make sure their robot can execute a little dance by reading specific instructions in an external memory. Their basic execution and code structure is somewhat similar to a Java Virtual Machine. This is also an example of software architecture. Students should now realize that this course follows a bottom-up approach. The control of each individual hardware element is encapsulated in C++ class or function in a library and we can reuse code to fulfill more complex needs.

5. THE CHALLENGE AT THE END

Up until this point in the semester, assignment on a weekly basis has been the operating mode. For the remaining weeks (about four), the course will turn into a real project. Students will receive a much longer description of what the robot is expected to do. They will also get sensors to install on the robot. Teams will have to understand this challenge, evaluate how they can bring an appropriate solution, make a plan to develop the code, manage the unexpected problems and test the solution. In other words, a real short project. Naturally, most of the concepts covered previously will have to be reused but in complex situations this time.

The robot has to follow a special kind of race. To design this course, a table (4'X8') made of white melamine on top (figure 2) is used. We add black tape to mark this course and we complete with various obstacles: aluminum posts, acrylic walls, magnetic or light sources, etc. This robot race changes every semester. One, two or three robots can be on the table at the same time depending on the challenge and what is the desired robot interaction. Usually, if more than one robot are on the table at the same moment, they have to communicate using IR emitters/receivers similar to those found in TV remote controls.

The challenge is to be able to guide the robot to follow a certain path using a line tracker which can distinguish the black and the white of the surface while other sensors are used to identify objects in the surroundings. Students have to analyze how various sensors work and which electronic resources inside the microcontroller should be used at the precise moment. Obviously, once the robot is put on the table and is moving, most teams have to re-evaluate their strategy based on the results they observe. This challenge offers an opportunity for students to develop their conception and creativity skills, even in a first-year project where computer and software technologies are part of a complex system.

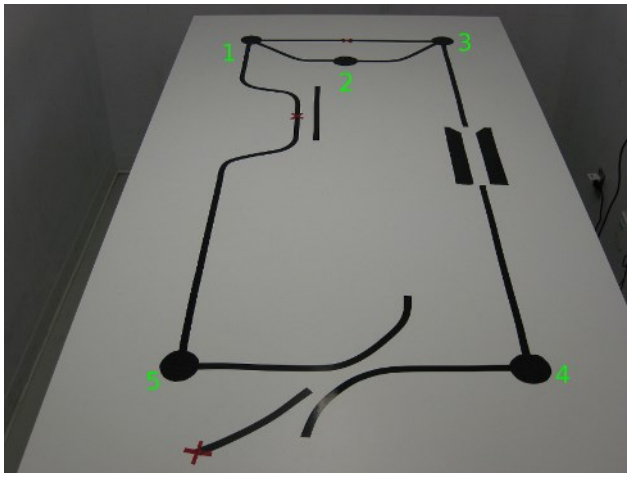

Figure 2. Example of a robot course

The last day is reserved for a public presentation. The table used by students will be moved in the middle of a large public area. The robot performances will be evaluated by judges. Students will prepare a poster session

at their kiosk as well. Judges will also evaluate this oral test. Visitors like to pass and to watch what the robots have to do so the interaction with people is interesting (figure 3). To conclude, we present a trophy to the best teams. It represents an additional source of motivation for students.

Figure 3. Public presentation

6. EVALUATION

This project course is formed around many different activities. Thus, the diversity is also found in the evaluation. First, the robot assembly is not evaluated. Students know that they need this platform for the duration of the semester so we prefer to give a general assessment on the general quality for this activity, especially on the soldering aspects.

Students will also have to submit three shorts programs during the semester for evaluation. They also receive a written report regarding the general quality of their code by the teaching assistants. Submission is done through SVN. To get a correct submission can be a problem at the beginning but it also helps students to work with the system. Even in a project, it still makes sense to have an exam to measure each student to make sure everybody is capable of some contribution to the teamwork and understands the key concepts. Their library (week #8) is also evaluated and a report has to explain how they have made it.

At week number 9, teams have an interview with the teacher where different questions will be asked. The code they have written will be reviewed and technical questions will be asked about what the code does. Usually, these questions are addressed to one student at a time. Some non-technical questions will also be asked about the team it-self: how the team members communicate, how they organize their work sessions, how they view the project course so far, etc. Thus, this turns into an assessment by the teacher. He can give feedbacks to the members based on his observations and answers from the team. The teacher usually has a good idea after an interview if a team will need more attention during the remaining weeks or if it is in a good position to succeed.

The challenge is evaluated by the teaching assistants during the public presentation with a scoring rubric and it is based strictly on the behavior of the robot and its capacity to perform well during the robot race. The poster session is also evaluated with a scoring rubric.

7. SUPPORT

A first-year project including hardware assembly and low-level programming requires a lot of direct support to students and good equipment in a well-organized laboratory to be successful. Early investments in quality hand tools, multi-meters, oscilloscopes and soldering irons are very important. Even small lockers for robots or for the teaching assistant's special tools have to be considered. The design and the refinement of the robot over the years are even more important to reduce the number of repeated minor problems. Otherwise, students pass more time on these little problems and less on the important concepts. First-year students will mature in their debugging skills over the next few years but they are limited at the beginning.

A carefully structured course web site is also very important. The site for this project is maintained continuously. It includes pictures, videos, assembly instructions, references to various external sites, datasheets, step-by-step debugging procedures, advices, and many more. Presented at the right moment and gradually during the semester, their impact is important and they give confidence to students that they have the first line of support they need.

Recruiting the best students to become teaching assistants is a good strategy. These assistants become the second direct line of support when students have questions. A part-time engineer, member of the technical staff, helps a lot in the preparation of this course: parts orders for the robot kits, equipment in the lab, questions from students, long term improvements, etc. The teacher has to play his role as well. He will spend less time in front of the class on PowerPoint presentations and more helping students with their robots. However, his direct support will make a difference for the success of his students.

8. RESULTS

This project course was evaluated by 43 out of the 51 registered students of the fall semester of 2013 with the standard project evaluation form used at Polytechnique Montréal. This semester was particularly important to us because some significant modifications to the course web site and to the robot it-self were completed in August of the same year. Table 1 presents the results to some questions pertaining to the course structure and indirectly to the support offered to students. These results demonstrate a very good appreciation by the students. Results for previous semesters were similar.

Table 1: Project course evaluation results – fall of 2013.

	Perception Results			
	Disagree		**Agree**	
Evaluation questions	**- -**	**-**	**+**	**+ +**
The teacher has paid attention to team aspects	1	2	8	32
The skills development was in accordance with the project objectives	0	1	4	38
Workload is well distributed throughout the semester	0	3	11	29
Final mark was based on various aspects	0	1	8	34
The level of difficulty is appropriate for this project	1	2	14	26
The project is well organised	0	1	7	35

9. CONCLUSION

After many readjustments over the years, this first-year project has stabilized. Many early problems were about the support and the minor details that, when all added together, were too time consuming. However, we conclude that it was worth fixing all these problems. The impression is that we now spend more time on important learning and team experiences.

Starting with sort programs and using them as bricks to build a complete system is also seen by students as valuable experience. It shows how small details add up and how team members have to interact to develop a complete solution to complex problems. The variety of activities proposed by this first project-based course gives a good foundation for the next three project based courses at Polytechnique.

Acknowledgements

The author would like thank the Department of Computer and Software Engineering of Polytechnique Montréal for its support of this project over the last 9 years, especially Michel Dagenais, Yves Boudreault and Laurent Tremblay. Matthew Khouzam was a main designer of the robot and the author wants to thank him as well.

References

[1] Yves Boudreault, " Environnement favorisant l'apprentissage des concepts fondamentaux de la programmation ", Actes du 4e colloque annuel de DIVA, (Montréal, QC, 17-18 May 2005), 8 pp., 2005.

[2] Jérôme Collin, INF1995 project course [online], http:// http://www.groupes.polymtl.ca/inf1995/ fichiers/

[3] Matthew B. Crawford, *Shop Class as Soulcraft: An Inquiry into the Value of Work*, New York, NY: The Penguin Press, 2009, 256 pp. {ISBN: 978-1441800107}

[4] Barry S. Fagin and Laurence Merkle, "Quantitative analysis of the effects of robots on introductory Computer Science education", Journal on Education Resources in Computing (JERIC), vol. 2, Issue 7, 2002.

[5] Aaron Gage and Robin R. Murphy, "Principles and experiences in using Legos to teach Behavioral robotics", in Proceeding of the 33rd ASEE/IEEE Frontiers in Education Conference, (Boulder, CO, 5-8 November 2003), 6 pp., 2003. {ISBN: 0-7803-7961-6}

[6] IEEE Transaction on Education, Special Issue on Robotics in Education, February 2013, IEEE Education Society, Volume 56, Number 1, 147 pp. {ISBN: 18-9359}

[7] L. Major, T. Kyriacou and O.P. Brereton, "Teaching novices programming using robots", in Proc. EASE 2011 15th Annual Conference on Evaluation & Assessment in Software Engineering, (Durham, UK, 11-12 April 2011), 10 pp., 2011. {ISBN: 978-1-84919-509-6}

[8] F. Mondada, M. Bonani et al., "The e-puck, a Robot Designed for Education in Engineering", in Proc. 9th Conference on Autonomous Robot Systems and Competitions, (Castelo Branco, Portugal, 7-7 may 2009), 7 pp., 2009.

[9] Noam Nisan and Shimon Schocken, *The Elements of Computing Systems: Building a Modern Computer from First Principles*, Cambridge, MA: The MIT Press, 2005, 344 pp. {ISBN: 978-0262640688}