More Missing the Boat - Arduino, Raspberry Pi, and Small Prototyping Boards and Engineering Education Needs Them

Peter Jamieson
Department of Electrical and Computer Engineering
Miami University
Oxford, Ohio 45056
Email: jamiespa@miamioh.edu

Jeff Herdtner
Department of Electrical and Computer Engineering
Miami University
Oxford, Ohio 45056
Email: herdtner@miamioh.edu

***Abstract*—In this work, we describe a range of prototyping boards such as Arduino, Raspberry Pi, and BeagleBone Black, and we show how these devices are being used in our ECE curriculum in a range of courses for projects. We describe the continuing challenges we have with adopting such technology from an educational standpoint, and some best practices/techniques we have learned and adopted to include these devices in our courses. We believe integrating these devices into our course flow is of a huge benefit to both our curriculum and our students.**

I. Introduction

In 2011, we wrote a paper called "Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat?" that asked the question if electrical and computer engineering (ECE) was ignoring the growth of Arduino (arduino.cc) prototyping boards as a platform to teach aspects of ECE, and in particular, to teach embedded systems [1]. At that time, the Arduino world was showing large growth because the artist created community was allowing people from all walks of life to use a micro-controllers and electronics to implement all sorts of projects in a strong sharing community. This community differed significantly from the heavy front-end learning expected by engineers who used and supported traditional microprocessor boards. We felt that these devices had some place in traditional ECE education and we described how a course on embedded systems used Arduinos and the pros and cons of such devices.

The Arduino has continued to have success in the "Maker" world, and our curriculum has expanded the use of the Arduino to more than the embedded system course. Additionally, we have opened our embedded system design course to more than just Arduino boards and have seen students use Raspberry Pi (raspberrypi.org) and BeagleBone Black Boards (beagleboard.org). These new boards still have much of the open source and maker communities, but the devices themselves are more powerful allowing even more design possibilities.

In this paper, we describe what these devices are, and then, we describe how these devices are being used in our ECE curriculum in a range of courses. With our experiences, we describe the continuing challenges we have with adopting such technology from an educational standpoint and some best practices/techniques we have learned and adopted to use these devices in our courses. In particular, with the large wealth of open source projects available online, how do we assess projects to determine what students are building and learning. Additionally, we make claims as to why we believe integrating of these devices into our course flow is of a huge benefit to both our curriculum and the student.

The remainder of this paper is organized as follows: section II provides a background these prototyping devices and open source and how people have assessed projects. Section III takes a more detailed look at the prototyping devices, and section IV describes student projects that used them. Section V provides a description of best practices we have adopted for using these devices in courses. Section VI includes a short discussion and conclusion of this work.

II. Background

The three main areas we address in this paper are Project Based Learning (PBL), open source projects assessment, and prototyping boards used in ECE curriculum.

A. Project Based Learning

Project Based Learning (PBL) curricula (which is a version of Problem Based Learning and has a wide literature base [2]) is normal in many fields with examples in engineering , business, and medicine [3], [4], [5]. PBL pedagogy centers learning around the activity of the student. The approach focuses on building projects and allowing the student to learn on the fly as they face problems. Projects are spaced throughout the degree, hence the name PBL curriculum. The accreditation agency, ABET, among other entities, influenced engineering programs into including a major capstone around 1995 to 1997 [5]. For computer engineering curriculum, lab only courses [6], [7] slowly evolved to include both labs and final projects. The senior capstone has been studied to help understand how to prepare students for this culminating experience [8], [9].

Among the vast range of research on the impact of PBL on engineering curriculum (among other areas) there has been some focus on both the usefulness [10] and the assessment of projects (though assessment is still lacking). Much of the

research on assessment of PBL is lacking and most contributions are focused on describing the PBL course [11]. There are some academic reviews by Dym *et. al.* [3] and Graham [12].

B. Open Source Projects

Within the domain of PBL, two questions tend to arise: First, should projects be allowed to use the available plethora of projects on the web released as open source software (OSS) and projects (OSP)? Second, if OSS and OSP are allowed to be used, then how do we assess students contributions and learning?

The majority of literature that, partially, address these issues is from the OSS and computer science and software engineering education. In particular, many of the early documented attempts are in teaching software engineering use OSS [13], [14]. This includes using the most popular OSS project, Linux, to help teach operating systems [15], and includes senior capstone student work [16].

Assessment of these projects is not not deeply examined, but there are a few documents that talk to the issue. Nascimento *et. al.* go into the most depth with their study on the research issues of using OSS in courses [17]. In addition, Pedroni *et. al.* take a look at assessment techniques, but focus more on student perception of the value of the experience [18]. Unrelated to academic class assessment, Rigby *et. al.* [19] take a look at peer-review assessment of software, which is a common practice in software design, and has some ideas that could be used in a classroom setting.

C. Prototyping Boards

In electrical and computer engineering (among other majors) the prototyping board is a prefabricated board that includes a microchip or set of chips that allows various types of systems to be experimented with, designed, and tested without having to build the PCB (Printed circuit board) and test that part of the system. The word *prototyping* implies that the system, once, completed, could be designed more efficiently, but during prototyping most of the systems features can be used on these boards. These boards are used both in industry and education.

We do not provide a comprehensive list of such boards, but some common examples include: Altera's [20] FPGA prototyping boards such as Terasic's DE2 board that allow full systems to be implemented. Similarly, an open source FPGA related board called, NetFPGA [21], is being used to kickstart a prototyping board - ONetSwitch www.kickstarter.com/projects/onetswitch/onetswitch-open-source-hardware-for-networking. DSP boards, such as Texas Instrument's TMS320C6713 DSP Starter Kit (DSK) www.ti.com/tool/tmdsdsk6713, can implement DSP applications. Microprocessor boards, including the ones we will discuss in this paper (Arduino www.arduino.cc/, Raspberry Pi www.raspberrypi.org/, and BeagleBoard beagleboard.org, allow embedded systems to be prototyped. Ettus Research's USRP devices used in implementing Software Defined Radio systems, might also be considered as a form of prototyping board that has had an impact on education by allowing students to work in the radio domain [22].

Though these boards are used in undergraduate education, there is little discussion or evaluation on how to use them in courses. Pritchard and Mina do take a look at similar types of prototyping boards and classify them from three perspectives: hardware intensive, software intensive, and ease of implementation. Other work has focused on remote lab implementation, which includes a large body of scholarly work (we suggest a review paper [23]). Additionally, since microprocessors were available they have been included in labs and continue to be a major part of computer engineering [24]. PBL and prototyping boards make a good mix and multiple efforts have focused on this including robots embedded throughout the curriculum [25] [26], and FPGA boards for projects and learning digital logic [27], [28], [29].

The major question we have is with the availability of open source designs, cheap prototyping boards, and access to hobbyists and professional, how do we include these powerful tools in the modern ECE curriculum?

III. Arduino, Raspberry Pi, and BeagleBone Black ... Oh my

A. Arduino

"Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments" [30].

The UNO (the base system) consists of a microcontroller (an ATmega328 [31] microprocessor), a USB to serial chip, and an AC to DC power converter. The UNO can either be built by hand or can be bought premade from a seller such as sparkfun.com costing approximately 25 USD. The Arduino software platform is written in Java and is based, mainly, on Processing [32] (a language developed for artists). The IDE is installed on a machine and then can program the UNO over the USB. The base IDE includes a number of examples for blinking LEDs, making noises, etc. The UNO is only one type of Arduino board and many others exist in varying form factors.

We would classify the Arduino UNO as a simple microprocessor board that is easy to use for bit-banging based projects. Bit-banging is the fine grain manipulation and control of single bits or pins. The device is easy to learn to the point that the student can learn how to use the device with web resources alone without any formal instruction. This is because of the community that supports Arduino started with artists and these people provide significant help to beginners (as they are) as opposed to engineering forums for other microprocessors, which include responses such as, "go do your own homework/assignment". The ease of use and friendliness of the community does, however, bring a challenge to educators when assessing projects developed on these devices (which we will discuss later).

B. Raspberry Pi

"The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to

program in languages like Scratch and Python" raspberrypi. org.

The Raspberry Pi (we will call it just Pi) is a full-fledged computing system with a 900 MHz ARM based processor (www.arm.com/). The board can be loaded with the Linux and GNU tools, a monitor can be connected to it via the HDMI port, and input devices, such as a mouse or keyboard, can be connected to the USB ports. In other words, the Pi is a full computer that can be used for embedded system projects, but is also useful in the domain of implementing servers and other computer system applications. The additional processing power makes the Pi a much more powerful device, but accessing and using the general purpose input/out (GPIO) ports is more difficult and makes the Pi less useful as a starter device when controlling (bit-banging) simple electronic circuits. The Pi costs approximately 40 USD.

The Raspberry Pi community is also very active and helpful. However, a student needs to learn command line usage in Linux, programming tools in Linux, and other computing concepts/skills, which makes it harder to use for beginners to programming. Since the device is newer than the Arduino there are not as many open source resources, but there is more projects than one person can track meaning that assessment is equally challenging.

C. BeagleBone Black

"Makers, educators, explorers, professional engineers and corporations seeking to build upon a rich ecosystem are all encouraged to participate in BeagleBoard.org. With so much openness and thousands upon thousands of examples of people doing fun, interesting and profitable things with Beagles, you're missing out if you don't count yourself in for whom Beagle is intended. beagleboard.org

The BeagleBone Black (which we will call BBlack) provides functionality of both Arduino and Pi. The BBlack has a powerful processor (ARM processor at 1GHz) that can run Linux similar to the Pi, but the BBlack also has I/O capabilities to do simple bit-banging with circuits to the point that the BBlack can run the Arduino IDE (with a little bit of configuration www.logicsupply.com/blog/2014/09/24/tutorial-running-arduino-ide-beaglebone-black/.

The BBlack is also low cost and costs approximately 45 USD. The community is open and helpful, but is smaller than other two groups, but has the benefit of leveraging various projects that are designed for the other prototyping boards.

IV. Student Projects and Courses using these Boards

A. Courses using these boards

At Miami, the Arduino UNO has been used in a number of courses. In particular, the UNO is used in our introductory electrical and computer engineering course (ECE 102), our second course on circuits (ECE 303), our embedded systems design course (ECE 387), and optionally, in our computer organization course (ECE 289) and our senior capstone. In the introductory course and the circuit course, the department provides kits for the students, but for the other courses, students are expected to purchase the device in lieu of a textbook.

For projects in both our Senior Capstone and embedded system design, both the Pi and BBlack are common choices depending on the needs of the project. Again, these devices are purchased by the students both because of their low cost and student preference to have the device to play with and use outside their course work.

The amount of instruction provided on how to use these devices is very little. In the introductory course (ECE 102), the Arduino is introduced over two 55 minute lectures where the instructor shows how to connect an LED and pull-up switch. A basic program that shows how a light can be dimmed using pulse width modulation is shown to the students. From there, a student is expected to complete six labs using arduino and basic circuits to complete 3 assigned tasks:

- Knight Rider display which is a timed lighting of a series of LEDs
- Control a servo with a switch
- Build a simple ball catching system

and 3 student/TA created challenges that create a problem that needs to be solved using the device, some components, and some creativity.

In the circuit course (ECE 303), additional instruction is provided on the Arduino since some of the students come from outside the electrical and computer engineering department (for example, mechanical engineers), and they require a basic introduction to the board. Other than this, students are expected to learn how to program and interface their circuits with the Arduino by accessing tutorials and projects on the web.

For the Pi and BBlack, no formal instruction is provided, and students are expected to learn how to use these devices on their own. Computer engineering students will have learned command line Linux instructions and the compilation process in their computer organization course (ECE 289), but electrical engineers, who do not have this training, are expected to learn this as needed for their projects.

B. Student Projects

With these devices, students have created some impressive projects. The majority of these impressive projects are built in the ECE387 course and senior capstone design projects. For example, in ECE387 students created auto-tracking paintball guns, built a segway like device, built a bug-tracking system, and hypno-cubes. Videos and source code for many of these projects can be found at www.users.miamioh.edu/jamiespa/teaching.html, but as Google Code hosting service will be terminated in 2015, the availability of these design files will be for a limited time.

There are also a number of senior design projects that use these devices as the microprocessors in the student's designs. For example, in 2015 a group of students is building a microphone array that can distinguish people location in a room based on conversation. In 2014, a group of students used vibration sensors and a Raspberry Pi to triangulate the location of people based on their foot steps. These are just

a few examples of complex systems that students can create with these devices.

Lastly, the IEEE student group commonly uses these devices for their projects which includes a 60 inch spectrum analyzer (built with CPU fans, foam chips, and led lighting) and a coin operated arcade game emulator. The availability of these boards has vastly expanded the projects that the student run group can do.

V. Challenges with adopting prototyping boards

As great as these microprocessor prototyping boards are based on functionality and low cost, including them in courses has a number of challenges. In particular, the most difficult challenge is with how to deal with the availability of open-source projects and shields and libraries, and then assessing what a student has done. We will describe what our current best practices are, but by no means have we solved the problem of assessment and project creation completely.

A. Assessing Projects with the Availability of Open-Source

We have been using Arduino UNO in classes since 2010 and were one of the earlier adopters in this community. In 2010, there was a significantly large community of users and open source projects, but since then there has been a continuous increase of projects and people. The number of Arduino shields and supported libraries, where a shield is an easily connectable separate device that can be attached to an Arduino and allows interfacing with another chip (for example a motor drive shield), has also increased significantly making it possible to use complex chips without students having to understand much on interfacing. Additionally, libraries that support both shields and interfacing with sensors, actuators, and other chips are continuously being improved. All of this means that there is a massive amount of information, code, and designs that students can leverage in creating their own designs. The question remains, how can we assess the idea of "their project" and verify that students are learning.

One simple solution is to move towards first principles and push students do do low-level interfacing without libraries. Arguably, this approach should be done at least once since it helps student's understand some of the details in interfacing. However, forcing students to reinvent the wheel for every device they will use is less useful since, arguably, in their industrial jobs the goal will be to leverage existing designs and code bases to create larger systems.

Another solution is to allow students to use any code base and assess the system based on the final product. In this approach, the instructor assumes that to create a complex system that a student will spend significant time understanding existing libraries, how those libraries can be used, and mixing more than one library/api together to achieve a complex task. We used this approach for a number of years, but we have begun to notice that some students are building projects that are similar to complete kits that can be purchased. For example, the hypno-cubes student's created in 2010 and 2012 can now be built with the purchase of a kit. For this reason, these types of projects are no longer considered an appropriate project.

The third approach we have used is to have the students deliberately prepare a document that shows what code they used and what they have added/designed. In this approach, we allow students to use any code, but they are required to show how their code is differentiated from the existing code. In this way, a student is required to describe how they created their system and used existing modules within their system. This process is similar to a diffing a code-base from the open-source base, but we require the student to illustrate this in a single organized document that allows instructors to easily see what the students have done and what was already available. The downside of this approach is that it requires the students to spend additional time organizing their design in the final deliverable. Also, this approach can make the students design appear simple, when in reality, to even get each device to interface requires significant time and learning even with an existing code base.

Of each of these approaches, the main question is what are the learning objectives. In the courses where we use these devices, the main learning objectives in the third year is a "create" (metacognitive and create) and in the introductory courses is a "use" (metacognitive and apply) in the Revised Bloom Taxonomy [33]. At these levels in the taxonomy, the question of what the student is learning is mainly related to the product, and all three approaches are sufficient for assessment. However, to promote creativity and reflection, we are moving towards our last approach in which students need to show their contributions differentiated from existing code-base and examples.

B. Project progress and success

Another significant challenge with projects, which is true independent of working with the above described prototyping boards, is scale of projects and ensuring student progress towards completing a successful project. In particular, students that transition into college think that a project can be completed in a single (long) evening, have a difficult time creating a project, and have very little skill at project organization. Additionally, students face many challenges in terms of working in group projects such as finding common meeting times, and dealing with non-contributing team members [34], [35], [36].

For project creation, we use one of two approaches; first, student's propose (orally and written) what they would like to do, which the instructor can modify, make suggestions, and provide fallback options, and two, a custom project is created that all students will do in the class. In both situations, the projects are successful based on the experience of the instructor and their understanding of what is doable in the time allotted.

To ensure progress we have tried to implement progress meetings, and these are successful for senior design projects, but they seem to have very little impact on early courses. We believe reality is that until a student experiences how difficult a project is to complete in a very short period of time, they do not learn to work continuously through a semester. As much as we provide direction, the learner has to experience how hard it is to get things built and working.

We also provide instruction on the use of design methodologies such as agile design [37], but this is not necessarily the focus of the courses the projects are in, and are ideas that tend to be taught later in the curriculum. Even with knowledge of these methodologies and best practices, students still have a

difficult time applying the techniques to improve the progress of their projects.

VI. Conclusion

In this paper, our goal was to expose educators to the benefits and challenges of using modern prototyping microprocessor boards - Arduino, Raspberry Pi, and BeagleBone Black. Each of these devices has been used in a number of our courses for student projects of many different levels. The main challenge with using these boards is identifying and assessing student projects based on the availability of open source designs and the community that supports these projects. We describe how we approach these problems, but we still do not believe we have perfected the use of these devices.

Overall, the students like the low cost of these devices and the ease of use that allows them to create significant projects. As instructors, the projects that are being delivered show that students are improving on system design and are delving into real engineering systems motivated by their own creativity. This we believe justifies adopting these boards in a curriculum with the risks described earlier.

References

[1] P. Jamieson, "Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat?" in *Proc. FECS*, 2010, pp. 289–294.

[2] J. W. Thomas, "A review of research on project-based learning," 2000.

[3] C. L. Dym, A. M. Agogino, D. D. Frey, and L. J. Leifer, "Engineering design thinking, teaching, and learning," *Journal of Engineering Education*, vol. 94, pp. 103–120, 2005. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.72.1593

[4] J. Macias-Guarasa, J. Montero, R. San-Segundo, A. Araujo, and O. Nieto-Taladriz, "A project-based learning approach to design electronic systems curricula," *Education, IEEE Transactions on*, vol. 49, no. 3, pp. 389 –397, 2006. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1668283

[5] A. Dutson, R. H. Todd, S. P. Magleby, and C. D. Sorensen, "A review of literature on teaching engineering design . . ." *Journal of Engineering Education*, vol. 86, pp. 17–28, 1997. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.3949

[6] S. Areibi, "A first course in digital design using vhdl and programmable logic," in *Frontiers in Education Conference, 2001. 31st Annual*, vol. 1, 2001, pp. TIC –19–23. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.3048

[7] K. E. Newman, J. O. Hamblen, S. Member, T. S. Hall, and S. Member, "An introductory digital design course using a low cost autonomous robot," *IEEE Transactions on Education*, vol. 45, pp. 289–296, 2002.

[8] C. J. Lesko, Jr., "Building a framework for the senior capstone experience in an information computer technology program," in *Proceedings of the 10th ACM conference on SIG-information technology education*, ser. SIGITE '09, 2009, pp. 245–251. [Online]. Available: http://doi.acm.org/10.1145/1631728.1631804

[9] J. Goldberg, "Preparing students for capstone design [senior design]," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 28, no. 6, pp. 98 –100, 2009. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5335729

[10] R. A. Atadero, K. E. Rambo-Hernandez, and M. M. Balgopal, "Using social cognitive career theory to assess student outcomes of group design projects in statics," *Journal of Engineering Education*, vol. 104, no. 1, pp. 55–73, 2015.

[11] L. Helle, P. Tynjälä, and E. Olkinuora, "Project-based learning in post-secondary education–theory, practice and rubber sling shots," *Higher Education*, vol. 51, no. 2, pp. 287–314, 2006.

[12] R. Graham, "Uk approaches to engineering project-based learning," *White Paper sponsored by the Bernard M. Gordon/MIT Engineering Leadership Program. http://web.mit.edu/gordonelp/ukpjblwhitepaper2010.pdf*, 2010.

[13] D. Carrington and S.-K. Kim, "Teaching software design with open source software," in *Frontiers in Education, 2003. FIE 2003 33rd Annual*, vol. 3. IEEE, 2003, pp. S1C–9.

[14] S. S. Gokhale, T. Smith, and R. McCartney, "Integrating open source software into software engineering curriculum: Challenges in selecting projects," in *Proceedings of the First International Workshop on Software Engineering Education Based on Real-World Experiences*. IEEE Press, 2012, pp. 9–12.

[15] R. Hess and P. Paulson, "Linux kernel projects for an undergraduate operating systems course," in *Proceedings of the 41st ACM technical symposium on Computer science education*. ACM, 2010, pp. 485–489.

[16] K.-B. Yue, Z. Damania, R. Nilekani, and K. Abeysekera, "The use of free and open source software in real-world capstone projects," *Journal of Computing Sciences in Colleges*, vol. 26, no. 4, pp. 85–92, 2011.

[17] D. M. Nascimento, K. Cox, T. Almeida, W. Sampaio, R. Almeida Bittencourt, R. Souza, and C. Chavez, "Using open source projects in software engineering education: A systematic mapping study," in *Frontiers in Education Conference, 2013 IEEE*. IEEE, 2013, pp. 1837–1843.

[18] M. Pedroni, T. Bay, M. Oriol, and A. Pedroni, "Open source projects in programming courses," in *ACM SIGCSE Bulletin*, vol. 39, no. 1. ACM, 2007, pp. 454–458.

[19] P. C. Rigby, D. M. German, L. Cowen, and M.-A. Storey, "Peer review on open-source software projects: Parameters, statistical models, and theory," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 23, no. 4, p. 35, 2014.

[20] Altera, "ALTERA at http://www.altera.com," 2010.

[21] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, "Netfpga–an open platform for gigabit-rate network switching and routing," in *Microelectronic Systems Education, 2007. MSE'07. IEEE International Conference on*. IEEE, 2007, pp. 160–161.

[22] S. Katz and J. Flynn, "Using software defined radio (sdr) to demonstrate concepts in communications and signal processing courses," in *Frontiers in Education Conference, 2009. FIE'09. 39th IEEE*. IEEE, 2009, pp. 1–6.

[23] L. Gomes and J. G. Zubía, *Advances on remote laboratories and e-learning experiences*. Universidad de Deusto, 2008, vol. 6.

[24] W. Albrecht, P. Bender, and K. Kussmann, "Integrating microcontrollers in undergraduate curriculum," *Journal of Computing Sciences in Colleges*, vol. 27, no. 4, pp. 45–52, 2012.

[25] R. L. Traylor, D. Heer, and T. S. Fiez, "Using an integrated platform for learning to reinvent engineering education," *IEEE Trans. Education*, vol. 46, no. 4, pp. 409–419, 2003.

[26] T. A. Roppel, J. Y. Hung, S. W. Wentworth, and A. S. Hodel, "An interdisciplinary laboratory sequence in electrical and computer engineering: Curriculum design and assessment results," *Education, IEEE Transactions on*, vol. 43, no. 2, pp. 143–152, 2000.

[27] D. Bouldin, "Impacting education using fpgas," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*. IEEE, 2004, p. 142.

[28] C. M. Kellett, "A project-based learning approach to programmable logic design and computer architecture." *IEEE transactions on education*, vol. 55, no. 3, pp. 378–383, 2012.

[29] A. J. Araujo and J. C. Alves, "A project based methodology to teach a course on advanced digital systems design," *WSEAS Transactions on Advances in Engineering Education*, vol. 5, no. 6, pp. 437–446, 2008.

[30] Arduino, "Available at http://www.arduino.cc," 2010.

[31] *8-bit AVR Microcontroller*, ATMEL, 2011. [Online]. Available: http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf

[32] C. Reas, B. Fry, and J. Maeda, *Processing: A Programming Handbook for Visual Designers and Artists*. The MIT Press, 2007.

[33] L. W. Anderson, D. R. Krathwohl, and B. S. Bloom, *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Allyn & Bacon, 2001.

[34] B. A. Oakley, D. M. Hanna, Z. Kuzmyn, and R. M. Felder, "Best practices involving teamwork in the classroom: Results from a survey of 6435 engineering student respondents," *Education, IEEE Transactions on*, vol. 50, no. 3, pp. 266–272, 2007.

[35] V. Pieterse and L. Thompson, "Academic alignment to reduce the presence of 'social loafers' and 'diligent isolates' in student teams," *Teaching in Higher Education*, vol. 15, no. 4, pp. 355–367, 2010.

[36] M. Borrego, J. Karlin, L. D. McNair, and K. Beddoes, "Team effectiveness theory from industrial and organizational psychology applied to engineering student project teams: A research review," *Journal of Engineering Education*, vol. 102, no. 4, pp. 472–512, 2013.

[37] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, 2003.