Making the Move from C to Python With Mechanical Engineering Students

Dr. Burford J. Furman, San Jose State University

Burford "Buff" Furman has been on the faculty in the Department of Mechanical Engineering at San José State University since 1994. Prior to coming to SJSU, he worked at IBM in the Silicon Valley (San José, California) in the development of disk drive actuators and spindle motors. He has also worked as a consultant in the optomechanical and laboratory automation industries. His areas of teaching and research are primarily focused in mechatronics and solar-powered automated transportation.

Dr. Salman Ahsan, San Jose State University

Currently Salman Ahsan is an educator and mentor to young people he teaches part-time at San Jose State and Seattle University. He is also working on a services company that specializes in the artificial intelligence and machine learning space. In the past he worked in the semiconductor industry, in companies like Linear Technology (now Analog Devices Inc) and Maxim Integrated. Salman studied at the University of Pennsylvania (B.S.E), Princeton University (Ph.D) and University of California at Berkeley (M.B.A).

Mr. Eric Wertz, self/EduShields

Eric Wertz is a software engineer most recently involved in embedded systems and education. He has been a volunteer educator developing both hardware and software course materials for the Mechatronics program at San Jose State University (SJSU) for more than a decade. He has been involved in operating systems and driver development, mobile computing device development, DevOps and security, and authentication and encryption toolkits. He holds a BS in EE/CS from the University of California at Berkeley.

Making the Move from C to Python With Mechanical Engineering Students

Abstract

Work is underway in the Mechanical Engineering Department at San José State University to transition the first course in computer programming (ME 30 Computer Applications) and a follow-on course, ME 106 Fundamentals of Mechatronics, from C to Python. Both courses make extensive use of a microcontroller to teach the fundamentals in both subjects, and heretofore have used the C language and the Arduino platform, but now both courses have moved to Python and to the Adafruit Feather M4 Express board, which can run Python natively on its associated microcontroller. Prior to the transition to Python, ME 30 had a relatively high failure rate between about 10 - 35%. Since transitioning to Python, the failure rate dropped dramatically to about 3% in the fall of 2019. The paper will outline the previous structure of the courses, explain the motivation for transitioning from C to Python, and discuss the pros and cons of the transition observed to date.

Introduction

In a paper written ten years ago, Furman & Wertz (2010) described ME 30 Computer Applications, a two-unit required course for mechanical engineering students at San José State University. For most students, ME 30 is the first (and last) computer programming course per se that they take in the ME program. ME 30 serves primarily as a prerequisite for ME 106 Fundamentals of Mechatronics, which is a three-unit (but perceived by students as being more like five units) required class that covers the integration of electronics, mechanics, and software that occurs in mechatronic systems (Hsu, 1995; Furman, et. al., 1996). ME 30 is typically taken by freshman or sophomores, and is the only regular course in the ME program with no prerequisite. Consequently, some students may not have had a first course in calculus or physics at the time they take the course. Around 2010, changes were made to ME 30 to introduce the use of the Arduino microcontroller and a custom-designed add-on board to make the course more interesting to mechanical engineers and to prepare them more adequately when they encountered the same hardware in ME 106 and needed to design, build, and control a mechatronic system (Furman & Wertz, 2010). Appendix A lists the topics covered in lecture and lab for ME 30, and Appendix C lists the topics covered in lecture and lab for ME 106.

Until changes were made in the fall of 2018, students in ME 30 were taught the essential aspects of procedural programming using the C language. The course also included a taste of Matlab and a little bit of advanced charting using Excel. There are many good reasons to use C as a first programming language for mechanical engineers (Cheng, 2019; 2009; Furman & Wertz, 2010), and especially when it comes to interfacing to microcontrollers in mechatronic systems, however many academics assert that Python is a better programming language for beginners to learn (Luxton-Reilly, et. al., 2018; Ateeq, et. al., 2014; Jayal, et. al., 2011; Loui, 2008; Enbody, et. al., 2009; McCane, 2009; Mannila & de Raadt, 2006; Radenski, 2006; Rufinus & Kortsarts, 2006).

The problem and proposed changes

Before the fall of 2018, ME 30 had a relatively high failure rate, one of the highest in the department, where failure is defined as a student receiving a final grade of less than C-. Failure also includes 'unauthorized withdrawal', which is designated on the transcript as 'WU'. (A grade of WU is usually given when a student stops coming to class and turning in assignments). Figure 1 below shows the percentage of students who received D, F, or WU grades since 2008:

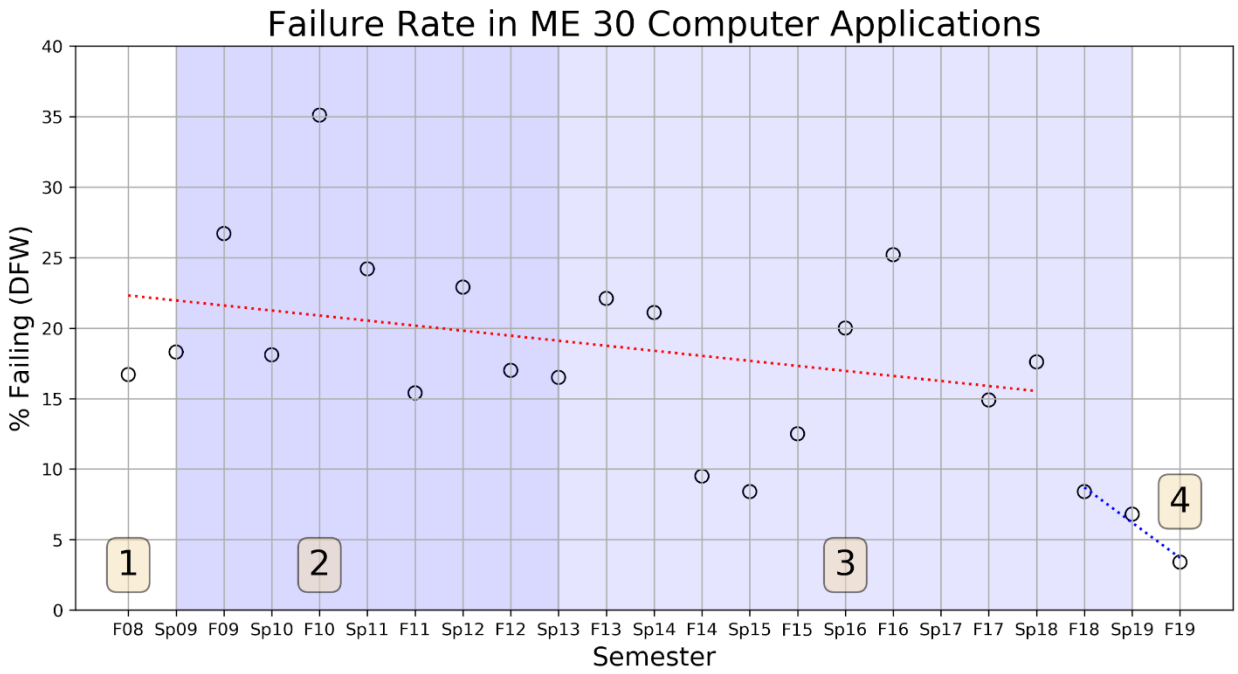

Figure 1. Historical failure rates in ME 30. The average failure rate from fall 2008 to spring 2018 was 19.1%. Data for spring 2017 was not available. Prior to spring 2018, C was the language used to teach procedural programming concepts in ME 30. From fall 2018 to the present, Python is the language used. The boxed numbers correspond to the four instructors who have taught the course from the fall of 2008 to the present, and the shading indicates the semesters they taught the class.

In 2017-2018, the University provided modest stipends for faculty teams to redesign courses with high DFW rates, and a team of faculty and students took up the challenge to improve the failure rate in ME 30. The team proposed two broad changes to be implemented in the fall of 2018:

1. Remove the lecture and lab modules covering highlights of Matlab and advanced Excel charting.
2. Change the programming language from C to Python.

The brief coverage of Matlab and Excel had been added to ME 30 beginning in fall 2009, because around that time, changes in the Introduction to Engineering course taken by all undergraduate engineering students had removed Matlab from its laboratory curriculum, and no other course in the ME program was covering Matlab. Similarly with Excel, it was observed by one of the authors that freshman students were often weak in basic (let alone advanced) Excel charting skills, and since no other course in the ME program was giving them the skills in Excel, it seemed to make sense to cover it in ME 30. The redevelopment team felt that by removing the Matlab and Excel components, several lecture periods and two lab experiments would be freed up, and it was surmised that this would allow the students to focus on ***one*** programming language instead of spreading their precious academic weeks to try and learn three different languages.

The decision to change from C to Python was made for a variety of reasons (Sobral, 2019). First, Python is recognized by many to have a simple syntax, dynamic typing, and language-like structure that makes it very accessible to those with no prior programming experience (Chapman & Irwin, 2015; Bogdanchikov, et. al., 2013; Agarwal & Agarwal, 2006; Radenski, 2006; Donaldson, 2003; Zelle, 1999). These aspects of Python free the instructor to be able to focus more on problem solving and programming concepts and less on syntactical and language specific details. Second, there is a tremendous amount of tutorials and resources available to learn how to program in Python, thus students have a wide variety of options to get help beyond the lecture, lab, and textbook (For example, Arora, 2018). Third, Python's versatility and widespread use beyond the ME academic program is attractive to students as more and more job postings list knowledge of Python as a skill that a qualified candidate should have. Fourth, and also significant for the 'downstream' effects on ME 106, there are now options for programming microcontrollers directly in Python (Bell, 2017; Plamauer & Langer, 2017; Tollervey, 2017; George, et. al., 2016; Rembor, n.d.).

Making the switch to Python

The switch to Python in the fall of 2018 in ME 30 was complicated, in part because of the integration of hardware (the Arduino) in the lab component of the course AND the need to almost simultaneously make major changes in ME 106 (which also had been based on the Arduino), because as soon as the change to Python was made in ME 30, the pipeline of students entering ME 106 would begin filling with students who did not know C, and knowledge of C is needed to program the Arduino.

We considered many options for an alternative to the Arduino: the Raspberry Pi, the BeagleBone Black, the ST Nucleo, the Micropython pyboard, and a series of boards from Adafruit Industries capable of running CircuitPython, a fork of Micropython. But with time running short to carefully evaluate alternatives AND re-write the lecture and lab curriculum over the summer of 2018 to be ready for the fall, we decided to continue with the Arduino in the ME 30 lab, but use Pymata, a Python library for the Firmata protocol (MrYsLab, n.d.) that allows a Python program

running on a host computer to communicate with the Arduino over a USB virtual serial COM port. One reason to use Pymata is that its extensions to Firmata allowed us to have both rotary encoder and servo motor functionality, which we needed, but which is not present in Standard Firmata. Doing so bought us time for evaluating an alternative microcontroller and also allowed us to continue to use the custom-designed Arduino shield that we had developed over the previous eight years. Appendix B gives details of the custom shield (called the YouKnow board) used through the spring of 2019. We chose to use the open-source Anaconda distribution (https://www.anaconda.com/) of Python for our lab computers, and recommended that the students do the same for their own personal computer, as it seemed to be the least complicated way to set up a full-fledged environment for programming in Python.

We also chose to have the students do their Python programming in JupyterLab, a web-based, interactive graphical interface and development environment that allows for relatively seamless integration of live code, text, and graphics all in one platform (Project Jupyter, n.d.). JuypterLab is available as an installable package, but it also comes installed with Anaconda.

The Arduino-Pymata approach to integrating hardware into ME 30 sufficed for the fall 2018 and spring 2019 semesters, however it was not an ideal solution, because to work with the Arduino hardware outside of the lab, students had to install Anaconda *and* the Arduino Integrated Development Environment (IDE) on their personal computer. With all the permutations of personal computing hardware and operating systems, getting all the software installed *and* working was not a slam-dunk for all students. We also ran into some trouble getting Pymata to work as expected and eventually figured out that we had to downgrade one of the libraries in the Python installation (the Tornado library) before the host computer would talk successfully to the Arduino. Nevertheless, we were pleasantly surprised to see at the end of the fall 2018 semester that the failure rate in ME 30 had plummeted to 8.4% (see Figure 1). The failure rates for the following two semesters also seem promising that the changes to ME 30 have made a difference and have significantly improved student success.

After the spring 2019 semester, we decided to make the switch from the Arduino to the Adafruit Feather M4 Express (FM4E) board (https://www.adafruit.com/product/3857). We chose the FM4E for a variety of reasons:

- Adafruit supports a port of Micropython (called CircuitPython) that runs directly on the FM4E.
- The FM4E comes with Circuit Python installed, and can be programmed by simply dragging a .py source code file from the user's host computer to the file system on the FM4E.
- CircuitPython has numerous libraries that support sensors, displays, motors, user interfaces, etc.

- The FM4E can be used with the Arduino IDE (with programs written in C), making it behave almost identically to an Arduino
- The price of the FM4E ($23) is reasonably close to that of a "genuine" Arduino
- The FM4E has a feature set and performance that can be leveraged by follow-on mechatronics courses, especially ME 106.

The change to the FM4E entailed rather significant changes to our custom-designed add-on board, because the Feather form factor is entirely different from that of the Arduino, and we could no longer use the Edushields YouKnow board, designed for Arduino Uno R3-compatibility. Appendix B gives details of the revised hardware for ME 30 using the FM4E.

Also in the fall of 2019, we revamped the lecture and the lab for ME 106 to use the FM4E instead of the Arduino as mentioned earlier. The content of the lab experiments stayed largely the same, but adjustments were necessary in some of them. For example, the FM4E is a 3.3 V device, whereas the Arduino is a 5 V device. This meant that for some sensors and things like motor drivers that had worked well with the Arduino, instead worked marginally or not at all at 3.3 V with the FM4E. In some cases, we needed to use voltage level-shifter devices and two supply voltages (5 V and 3.3 V) to get everything to work. The FM4E is also much more 'delicate' than the Arduino when it comes to sourcing and sinking current through its pins. The Arduino can handle up to 40 mA per IO pin and voltages between -0.5 V and 5.5 V, whereas the FM4E can handle up to 15 mA source or sink, and voltages are limited to within -0.6 V and 4.4 V with respect to ground. The voltage and current limitations with the FM4E are not of concern for ME 30, because students only use the rather self-contained Feather Hardware Kit and do not need to interface the FM4E beyond what is included with it. In contrast, students in ME 106 must build many circuits on solderless breadboards and interface them to the FM4E. We had several students 'fry' their Feathers by inadvertently applying voltages that exceeded the absolute maximums. However, on the upside, since the current handling capability of the FM4E is significantly lower than that of the Arduino, ME 106 students need to be more thoughtful about the design or selection of an interface to a device that requires significant power rather than trying to drive the device directly from a pin on the microcontroller. Understanding how to interface to devices requiring significant power is a learning objective in ME 106. The Arduino, with its relatively higher current capability, tends to let the students 'off-the-hook' in this regard and give the impression that everything in a mechatronic system can be directly connected to and driven by a microcontroller, which is not the case!

Outcomes so far

We surveyed and interviewed students in both ME 30 and ME 106 to get their take on the use of Python in these classes. In ME 30, a student survey was conducted at the end of the fall 2019 term to gather student sentiment on a range of subjects including the decision to use Python as the introductory programming language. Out of 42 respondents, 45% said that their interest in the class was enhanced by the fact that Python was used, while only 2% said that they would

have preferred a different language. The rest were neutral about the choice of programming languages for the class. Another question probed the students' interest in continuing to learn more about computer programming. Out of 42 respondents, 40% of the respondents said they were intrigued to learn more about computer programming as a consequence of taking this class. Only 5% said they had no interest in programming and have no intention to learn more. There were only a handful of students, 7 in total, who had some prior experience with either C or C++, and of those, four students found Python to be easier, two thought it was about the same and one thought it was harder.

Of particular note were individual interviews of two students (subsequently referred to Student 1 and Student 2) who had failed ME 30 when it was taught using C, and then subsequently passed ME 30 after it had changed to using Python. Student 1 failed the course in fall 2017 with a WU grade and then passed in fall 2018 with an A. Student 2 also failed ME 30 in fall 2017 with D-, and then passed in spring 2019 with a B. We would like to have had a larger sample of students who experienced both versions of the course, but sadly, these two were the only ones that could be identified from cohorts who had failed the C version and passed the Python version of ME 30 at the time of writing this paper. It is possible that we will find others over the next few semesters if they have delayed re-taking ME 30 at SJSU. Table 1 summarizes responses from the two students to five questions about their experience in ME 30.

Table 1. Results from interviews of two students who failed ME 30 when it was taught in C and subsequently passed ME 30 after it had changed to Python.

Question	**Student 1**	**Student 2**
How would you describe the *level of difficulty* learning to program in Python (compared to learning to program in C)? Indicate the level of difficulty using a number between 1 and 5, where 1 = Much easier, 3 = About the same, 5 = Much harder	1	2
How would you characterize your *interest* in learning how to program a computer in general? Indicate your level of interest using a number between 1 and 5, where 1 = Not interested, 3 = Neutral, 5 = Very interested	4	4
Did you have any programming experience prior to taking ME 30 (the first time)? Yes or No	No	No
How would you characterize *the use of a microcontroller* in helping you learn how to program in Python (or C)? Indicate your thinking using a number between 1 and 5, where 1 = Did not help, 3 = Neutral, 5 = Very helpful	5	5

To what degree did *removal of the content* of Excel and Matlab from ME 30 affect your success in passing the class when you took it the second time? (Indicate the degree of effect using a number between 1 and 5, where 1 = Not much at all, 3 = Some, 5 = A lot)	3	5

Both students clearly thought that it was easier to learn how to program in Python compared to C. These observations are consistent with those noted by others (Ateeq, et. al., 2014). Both students took the Python version of ME 30 before we switched to the FM4E, so we do not have input on the relative impact of the switch from using Pymata and the Arduino to programming in Python direct on the Feather. We surmise, however, that the use of CircuitPython on the FM4E made it simpler and less frustrating for the students than dealing with Pymata and the need to install drivers for the Arduino. Both students noted that the removal of the brief Excel and Matlab coverage from ME 30 had some, to a lot of impact on their success the second time they took the course. Notwithstanding the small sample size, it seems reasonable that reducing the scope of coverage in ME 30 would give students more time to master programming concepts using one language (Python).

ME 106 students were also surveyed at the end of the fall 2019 semester. About 56% (n = 64) indicated that they had taken the prerequisite ME 30 at SJSU. Of the 36 who had taken ME 30 at SJSU, 23 of them had taken the C version and 13 had taken the Python version. Of the 13 who had taken the Python version, 69% of them noted that ME 30 had prepared them moderately to extremely well programming in ME 106. Fifty seven percent of respondents indicated that they had learned or used C/C++ prior to ME 106. About 69% of the respondents said they had a little to a great deal of prior experience with the Arduino (and presumably programming it in C) Of these, about 41% felt that programming in Python on the FM4E was easier in comparison to programming in C on the Arduino. About 54% thought that Python on the FM4E was about the same as programming in C on the Arduino. A relatively small number (5%) thought programming in C on the Arduino was easier than Python on the FM4E. Thus, feedback from this cohort seems to indicate that programming in Python on the FM4E is at least as easy as or easier to deal with than programming in C on the Arduino. This is especially notable in that a sizeable number of the students who took ME 106 in the fall of 2019 had prior C/C++/Arduino experience and relatively few (13 out of 64) had been prepared by the Python version of ME 30 before taking ME 106.

The failure rate in ME 106 has averaged 5% over the 14 semesters prior to fall 2019, with two instructors teaching (high of 11.9% in fall 2012 (instructor 1) and a low of 0% in spring 2017 (instructor 2)). All of the 14 semesters before fall 2019 used C and the Arduino. With only one semester's worth of data with Python in ME 106, it is premature to conclude that the change to Python is making a significant difference, but it is interesting to note that the instructor who

taught ME 106 in fall 2019 had no students fail. The 10 previous ME 106 classes he taught with C and the Arduino had an average failure rate of 5.5%.

Conclusions and next steps

Time will tell if the reductions in failure rate in ME 30 and ME 106 will stick, but so far, the results are encouraging. We continue to modify both courses to accommodate the new microcontroller hardware and to address shortfalls that we have noticed over the last few semesters. For example, while the fall 2019 ME 30 class had the lowest failure rate that we have seen in over 10 years, and this is likely influenced by the changes described above, we think that some of the students were able to 'game the system' that term which made the outcome better than it should have been. Some of the loopholes that the students were able to exploit were:

- To mitigate the policy of no late submissions, the students were allowed to drop one lab score and one homework score. The idea was to accommodate students who, due to sickness or other emergencies missed an assignment. Several students used this provision to skip what they perceived to be the harder labs or homeworks, thereby robbing themselves the opportunity to learn some key concepts.
- To emphasize the 'learn-by-doing' method of programming pedagogy, the labs and homeworks were given larger weight vis-a-vis quizzes and exams. It became clear towards the end of the term that students were relying heavily on each other and looking for solutions to problems on the web to complete lab and homework assignments. Many students with stellar lab scores did not do so well on the tests.

These loopholes have been closed for spring 2020, so we look forward to the data on student performance that we will collect in the future.

Another step we would like to implement is to provide more examples and patterns of solid programming techniques especially in ME 30. Guzdial (2015) raises the question about the best way for students to learn to program and suggests that providing stronger instructional guidance for beginners is better than minimal guidance and lots of self-guided problem solving.

In ME 30, we have found JupyterLab to be a useful tool for code exploration, development, and reporting, but its integration with external embedded-systems is still very rudimentary. It is most valuable in the portion of the course where students are using Python solely on a host computer. A second order benefit is that it can be used for both presentation of lecture content and interactive code demonstration instead of having to awkwardly switch back and forth from static content in PowerPoint slides to a live Python code window.

Finally, the transition to Python and the FM4E has resurfaced questions among the teaching team about the pedagogical approach and focus for both ME 30 and ME 106, which are the first two courses that support the mechatronics curriculum stem in the ME department. When we taught C with the Arduino, it was relatively straightforward to get into lower level details of programming the microcontroller. Such knowledge is certainly appropriate for students headed toward careers in embedded systems, but only a small fraction of our ME graduates find their way into positions where they need to write code that close "to the metal". Most of our graduates, if they spend any time coding at all, probably will program for higher-level applications such as test automation or data analysis. Python is better suited for these programming tasks, so our current thinking is that the curricular changes we have made are better aligned with the needs of our students.

References

Agarwal, K., & Agarwal, A. (2006). Simply Python for CS0. Comput. Sci. Coll., 21(4), 162–170. (Available: https://www.researchgate.net/profile/Krishna_Agarwal5/publication/262274614_Simply_Python_for_CS0/links/54de78c40cf296663786a2a1/Simply-Python-for-CS0.pdf

Arora, S. A. S. (2018, September 5). Learn Python: Tutorials for Beginners, Intermediate, and Advanced Programmers. Retrieved January 11, 2020, from https://stackify.com/learn-python-tutorials/.

Ateeq, M., Habib, H., Umer, A., & Rehman, M. (2014). C++ or Python? Which One to Begin with: A Learner's Perspective. In 2014 International Conference on Teaching and Learning in Computing and Engineering (pp. 64–69). (Available: https://www.researchgate.net/publication/271425337_C_or_Python_Which_One_to_Begin_with_A_Learner's_Perspective)

Bell, C. (2017). MicroPython for the Internet of Things: A Beginner’s Guide to Programming with Python on Microcontrollers. Apress. https://link.springer.com/book/10.1007/978-1-4842-3123-4

Bogdanchikov, A., Zhaparov, M., & Suliyev, R. (2013). Python to learn programming. In Journal of Physics: Conference Series 012027. (pp. 1-5). (Available: https://iopscience.iop.org/article/10.1088/1742-6596/423/1/012027/pdf)

Chapman, B., & Irwin, J. (2015). Python as a first programming language for biomedical scientists. In Proceedings of the 14th Python in Science Conference (SCIPY 2015). (Available: http://conference.scipy.org/proceedings/scipy2015/pdfs/brian_chapman.pdf)

Cheng, H. (2019, March 29). "C" as Part of a Mechanical Engineering Curriculum. Retrieved January 7, 2020, from https://www.asme.org/topics-resources/content/c-as-part-of-a-mechanical-engineering-curriculum.

Cheng, H. (2009). C for the course: what do you teach if the ME curriculum allows only 10 weeks to devote to computer programming?. Mechanical Engineering-CIME, 131(9), 50-53. Retrieved January 7, 2020, from http://iel.ucdavis.edu/publication/2009/ME.pdf

Donaldson, T. (2003, May). Python as a first programming language for everyone. In *Western Canadian Conference on Computing Education* (Vol. 547, p. 2015). (Available: https://www.cs.ubc.ca/wccce/Program03/papers/Toby.html)

Enbody, R. J., Punch, W. F., & McCullen, M. (2009, March). Python CS1 as preparation for C++ CS2. In Proceedings of the 40th ACM technical symposium on Computer science education (pp. 116-120). (Available: https://dl.acm.org/doi/10.1145/1539024.1508907)

Furman, B., & Wertz, E. (2010). A first course in computer programming for mechanical engineers. In Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (pp. 70–75). (Available: https://ieeexplore.ieee.org/document/5552091)

Furman, B. J., Hsu, T. R., Barez, F., Tesfaye, A., Wang, J., Hsu, P., & Reischl, P. (1996). “Laboratory Development for Mechatronics Education,” Session 1626, Proceedings of the ASEE Annual Conference, Washington, D. C., June 23-26, 1996. (Available: https://peer.asee.org/6155.pdf)

George, D., Sanchez, D., & Jorge, T. (2016). Porting of MicroPython to LEON Platforms. Data Systems in Aerospace. (Available: https://indico.esa.int/event/145/contributions/820/attachments/873/1052/06a_-_Micropython.pdf)

Guzdial, M. (2015). What's the best way to teach computer science to beginners? Communications of the ACM 58(2), 12-13. (Available: https://dl.acm.org/doi/pdf/10.1145/2714488)

Hsu, T. R. (1995, June). Mechatronics for Undergraduate Mechanical Engineering Education. In *1995 ASEE Annual Conference Proceedings* (Vol. 1, pp. 1312-1324).

Jayal, A., Lauria, S., Tucker, A. & Swift, S. (2011). Python for Teaching Introductory Programming: A Quantitative Evaluation. Innovation in Teaching and Learning in Information and Computer Sciences, 10:1, 86-90, DOI: 10.11120/ital.2011.10010086 (available: https://www.researchgate.net/publication/265478507_Python_for_Teaching_Introductory_Programming_A_Quantitative_Evaluation)

Loui, R. P. (2008). In praise of scripting: Real programming pragmatism. Computer, 41(7), 22-26. (Available: https://ieeexplore.ieee.org/document/4563874)

Luxton-Reilly, A., Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J., & Szabo, C. (2018, July). Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (pp. 55-106). ACM. (Available: http://repository.falmouth.ac.uk/3051/1/ITiCSE_2018__WG3.pdf)

Mannila, L., & de Raadt, M. (2006, February). An objective comparison of languages for teaching introductory programming. In Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006 (pp. 32-37). (Available: https://dl.acm.org/doi/pdf/10.1145/1315803.1315811)

McCane, B. (2009). Introductory programming with python. The Python Papers Monograph, 1, 1-18. (Available: https://ojs.pythonpapers.org/index.php/tppm/article/viewFile/101/105)

MrYsLab. (n.d.). MrYsLab/pymata-aio. Retrieved from https://github.com/MrYsLab/pymata-aio.

Plamauer, S., & Langer, M. (2017, April). Evaluation of micropython as application layer programming language on cubesats. In ARCS 2017; 30th International Conference on Architecture of Computing Systems (pp. 1-9). VDE. (Available: https://ieeexplore.ieee.org/abstract/document/7948548)

Project Jupyter. (n.d.). Retrieved January 11, 2020, from https://jupyter.org/.

Radenski, A. (2006). "Python first" a lab-based digital introduction to computer science. ACM SIGCSE Bulletin, 38(3), 197-201. (Available: https://dl.acm.org/doi/pdf/10.1145/1140123.1140177

Rembor, K. (n.d.). Welcome to CircuitPython! Retrieved January 11, 2020, from https://learn.adafruit.com/welcome-to-circuitpython/what-is-circuitpython.

Rufinus, J., & Kortsarts, Y. (2006). Teaching an Introductory Programming Course for Non-Majors Using Python. Information Systems Education Journal, 4 (104). (Available: http://isedj.org/4/104/ISEDJ.4(104).Rufinus.pdf)

Sobral, S. R. (2019). CS1: C, Java or Python? Tips for a conscious choice. In 12th annual International Conference of Education, Research and Innovation, Seville, Spain, 11th - 13th November 2019. Disponível no Repositório UPT, http://hdl.handle.net/11328/2924

Tollervey, N. H. (2017). Programming with MicroPython: Embedded Programming with Microcontrollers and Python. O'Reilly Media, Inc.

Zelle, J. M. (1999, March). Python as a first language. In Proceedings of 13th Annual Midwest Computer Conference (Vol. 2, p. 145). (Available: https://mcsp.wartburg.edu/zelle/python/python-first.html)

Appendix A - ME 30 lecture and lab coverage

Table A1 below lists the topics covered in lecture and lab. There is one, 50-minute lecture per week and one, 175-minute lab per week.

Table A1. Lecture and Lab topics in ME 30

Week	**Lecture Topic**	**Lab**
1	Enrollment, Course Organization., and Overview of Computers	
2	Computers and Programming	
3	Variables and types	Intro to Python and JupyterLab
4	Functions & style	Variables, Expressions, & Statements
5	Conditionals (Decision structures)	Functions
6	Fruitful Functions	Decisions
7	Iteration	Fruitful Functions
8	Strings	Iteration
9	Lists	Strings
10	Dictionaries	Hardware (Feather M4 Express)
11	Tuples	Dictionaries and Lists
12	Files	Term Project Proposal
13	Classes and objects	Files
14	Classes and methods	Data Analysis (collaborative challenge)
15	Review	

The authors are happy to share course materials for ME 30 upon request.

Appendix B - YouKnow Arduino shield and feather hardware kit descriptions

YouKnow Arduino Shield

Before the change to the Adafruit Feather M4 Express board, we used the Arduino with a custom-designed ‘shield’ (add-on board) initially called the Spartronics Experimenter, then later the Edushields YouKnow board. Figure B1 shows the YouKnow installed on top of an Arduino Uno compatible baseboard.

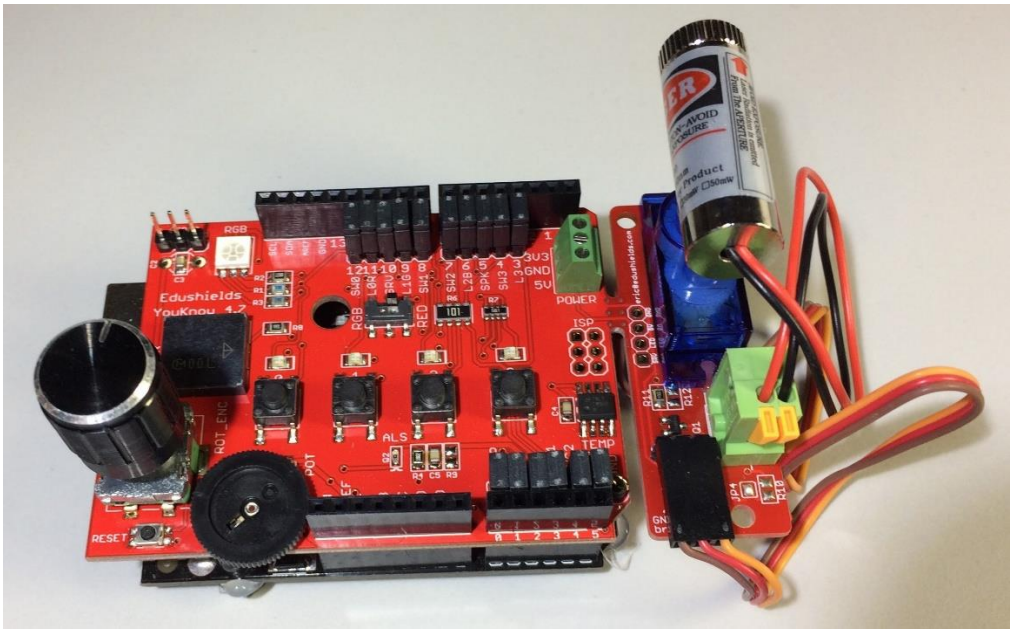

Figure B1. Edushields YouKnow 4.7 shield mounted on an Arduino Uno. A breakaway extension to the right has a mini-servo on top of which a laser module is mounted.

The YouKnow board has these features:

- Four tactile momentary buttons
- Four red LEDs
- One RGB (red ,green ,blue) LED
- One rotary potentiometer (for analog input)
- One rotary encoder
- One ambient light sensor (ALS) - phototransistor
- One LM34 10mV/°F temperature sensor
- One piezoelectric speaker
- One three-pin hobby servo header (powered from the 5V rail)
- One reset button
- One 5V/3.3V/GND screw-type terminal block
- One GND test point for scope/analyzer probe hook connection

- Sixteen I/O header pin jumpers
- One slide-switch for enabling RED LEDs vs. RGB LED
- One peephole for monitoring the Arduino on-board status LEDs from above
- 5V/3.3V compatibility with Uno-R3 format boards via the IOREF pin

Feather Hardware Kit (v1.0)

Changing to the Adafruit Feather M4 Express (FM4E) microcontroller board entailed designing an entirely new set of printed circuit boards (PCBs) to accommodate the Feather form factor. Figure B2 below shows the ME 30 Feather Hardware Kit.

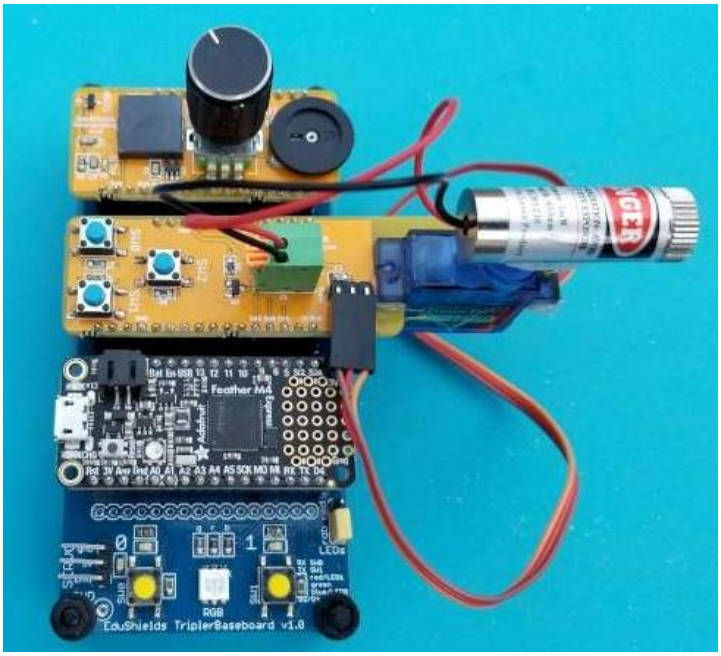

Figure B2. Edushields Feather Hardware Kit for ME 30. The Kit consists of (from the bottom of the figure toward the top) a 'Tripler Baseboard', the Adafruit FM4E board, the 'PewPewWing', and the 'InteractWing'. The Feather-based kit uses the exact same components as those used in the YouKnow shield, but with only two buttons/red-LED pairs versus the four pairs of the original.

The Tripler Baseboard is a PCB with three sets of female headers into which can be inserted any board with the Feather form factor. Also on the baseboard are two momentary pushbutton switches and an RGB LED. As shown in Figure B2, the FM4E occupies the first set of headers, and next above it is the PewPewWing (servo and laser pointer combo). The InteractWing (topmost board in Figure B2) contains a piezo-electric speaker for tone generation, a rotary encoder, a rotary dial potentiometer (continuously variable resistor), and ambient light sensor (ALS), and a 10mV/°C temperature sensor.

The PewPewWing unit is primarily intended to provide a stable mounting area for the hobby servo and laser pointer combo. This unit was designed for an exercise where students write a program to move the laser module right or left with two of the buttons and 'fire' the laser with the middle button while at the same time generating tones to simulate an arcade-like effect. Because the baseboard contains only two buttons and this exercise requires three, a third button was added (the other two electrically "overlap" the originals) but are grouped together for ease-of-operation.

The YouKnow board and the Feather Hardware Kit are available from co-author Eric Wertz (eric@edushields.com)

Appendix C - ME 106 lecture and lab coverage

Table C1 below lists the topics covered in lecture and lab. There are two, 50-minute lectures per week and one, 175-minute lab per week.

Table C1. Lecture and Lab topics in ME 30

Wk	**Lecture Topic**	**Lab**
1	Enrollment, course organization, review of basic electronics	
2	Signal sources	Soldering exercise
3	RC filters	Introduction to the Mechatronics Laboratory
4	Introduction to microcontrollers	Introduction to the Feather M4 Express
5	Programming the FM4E	RC filters
6	Diodes and transistors	Digital IO on the FM4E
7	MOSFETs	Photoresistors and transistors
8	Actuators and permanent magnet DC motors	Linear motion stage control
9	Midterm review and exam	Build your own servo
10	Op amps and amplifiers	Stepper motors

11	Comparators, DACs, ADCs	Electronic weight scale
12	Digital logic	Open lab for term project work
13	Serial communication	Open lab for term project work
14	Special topics	Open lab for term project work
15	Review, and ‘where-to-from-here’	Term project exhibition

The authors are happy to share course materials for ME 106 upon request.