

Developing a Creative K-12 Manipulative: An ECECS Capstone

Dr. Mike Borowczak, Erebus Labs

Mike is the chief scientist and founder of Erebus Labs - a Hardware Security and Engineering Outreach company located in Laramie, WY. He is also the Senior Data Scientist at a recently acquired startup. He has worked with university faculty to promote and extend K20 STEM outreach in Ohio, Oregon, Texas and Wyoming. He also has over a decade of industry and research experience - mostly revolving around the semiconductor and bio-informatics industries - with specific experience at Texas Instruments, Intel and Cincinnati Children's Hospital Medical Center. In addition to his industry experience, he has spent two years, while completing his PhD in Computer Science and Engineering, as a National Science Foundation GK-12 fellow - teaching and bring real-world STEM applications in two urban high schools. He has authored peer-reviewed articles, presented at national/international conferences, and taught undergraduate/graduate courses in both Hardware Security (computer science & engineering) as well as STEM Education and Outreach.

Dr. Andrea Carneal Burrows, University of Wyoming

Dr. Andrea C. Burrows received a Curriculum and Instruction: Science Specialization research Ed.D. from the University of Cincinnati, M.S. in Science Education from Florida State University, and a B.S. in Science Education/Biology from the University of Central Florida. She is an assistant professor in the Department of Secondary Education at the University of Wyoming, where she teaches courses in science methods, pedagogy, and research. Dr. Burrows also creates, implements, and evaluates grants at UW. Her research interests include secondary STEM partnerships and the meanings, negotiations, and conceptual changes associated with partnerships. She publishes and writes about STEM education extensively.

Developing a K-12 Computer Science Manipulative: An ECECS Capstone

Can you explain the basics of computing or computer science (CS)? Most computing experts have no problem talking to their peers about CS, but can they teach novices? Teaching and interacting with students without any prior scaffolding or exposure to CS concepts is outside the expertise area of most CS content experts and STEM faculty. This work highlights the need for, and current gap in K-12 computer science manipulatives. It focuses on the development and implementation of a solution that mitigates the traditional ‘experts teaching novices’ problem. The result, ‘A Block of Code’ allows students to visualize, manipulate and experiment with computer science concepts using a physical medium. This work looks at the impact of authentic value-added capstone projects on student’s soft skills by comparing results of a multi-year collaboration survey given to multiple senior capstone teams. The observed trends suggest that projects with community impact (irrespective of size or geographic constraint) foster increased communication, participation, and ultimately collaboration.

Introduction

There is a worldwide push to engage and develop K-12 student interest in Science, Technology, Engineering and Mathematics (STEM) disciplines¹. Some STEM collegiate programs, such as civil and mechanical engineering, seem to have a plethora of incoming and returning students. Why? Building blocks! One of the biggest recruitment disadvantages to the electrical and computing disciplines is the mystery behind their deliverables. What does an electrical engineer do? A computer engineer? A computer scientist? Few students ever have the opportunity to engage in authentic, age and cost appropriate activities related to the computing spectrum. In order to engage and develop student interest in computing professions we propose the need for low-cost physical manipulatives focused on computing rather than the construction, physics and mechanics in STEM². Finally, to reach wider student audiences, exposure to computing must go beyond traditional robotic applications³.

While developing interest in engineering and other STEM fields is paramount, developing well-rounded engineers is equally important. STEM professions have an image problem. The general public generally describes engineers as introverted and anti-social – “nerdy.” Among many K-16 STEM students, communication and other soft skills are perceived as secondary to content knowledge skills. Clearly educators, industry and ABET disagree – STEM professionals, even those in computing, need a balance of content and soft skills. Developing soft skills can occur in different forms, but assessment of these skills in non-intrusive yet authentic scenarios is only starting to occur. Studying the communication patterns of multiple capstone teams, given different types of problems to solve, allows us to develop insight into how real world problems can motivate a team while increasing collaboration and communication effectiveness.

Finally, while STEM and education faculty teach collegiate level content courses both faculties could benefit from richer collaboration and coordination. How can future K16+ educators develop a pipeline of STEM majors and graduates without partnerships between the faculties? We propose a capstone model in which teams report to a multi-disciplinary advisory panel rather than a single STEM advisor. This paper highlights how a single Electrical and Computer Engineering and Computer

Science (ECECS) capstone project can 1) be influenced by a societal need, 2) develop soft skills of a capstone team, and 3) create lasting mutually beneficial partnerships between academic faculties and external partners.

Purpose/Problem/Gap

Learning, development and concept synthesis can take many paths – natural learning progressions often being through the manipulation and unguided interactions with our environment. Consider a child approaching building blocks for the first time. There are no instructions – there are blocks and the laws of physics. Structures are built through trial and error. Manipulatives allow young children to learn, develop and then build simple structures without needing a civil engineer to scaffold their knowledge. Why should computing be any different?

The use of manipulatives in K-16 classrooms has already been well established. Research of manipulatives in fields that are more theoretical and abstract in nature, such as mathematics, are also being to emerge⁴⁻⁷. While specific areas of brain activity and development only occur during manipulative activities⁸⁻¹⁰, little academic instruction ever focuses on active manipulation¹¹. The problem in certain fields, in particular computing, is the lack of concrete, tangible and intuitive manipulatives. Pure computer science focuses on the theory of computing, algorithms, graph theory and so forth - translating foundational ideas from these realms into student manipulatives takes creativity, time, and effort to implement.

Some recent work has focused on the need to bridge the manipulation gap in STEM education¹², but to authors' knowledge there has not been any work in developing physical manipulatives specifically for Computer Science. Computer science is being pushed as a fundamental concept that should be taught to all children, along with the three R's of reading, writing and arithmetic. Even the President of United States is trying to allocate \$4 billion for the effort¹³. While many physical constructs have been promoted to aid in teaching CS to students, including robotics (e.g. Lego Mindstorms), and hardware/software systems (e.g. Arduinos and Raspberry Pis) these systems focus more on physical construction than computing, or are inherently abstract such that the amount of scaffolding needed to use the systems does not allow for an intuitive, inquiry learning experience. This work detailed in this paper focuses on the development of an ECECS project to create an intuitive, self-contained computer science manipulative for K-12 students. The end-product is a set of fifty 2x2x2 inch blocks that can be physically connected to produce and executable program.

Creating “A Block of Code”

The authors formulated the idea for “A Block of Code” after developing and implementing several dozen K-12 professional development (PD) workshop sessions focusing on STEM integration – especially in computing. Observations during each PD, in conjunction with teacher feedback led to a stark reality – teaching computer science through use of robotics diluted the message but teaching computer science using micro-controllers required a steep learning curve. In each instance, while teachers enjoyed the activities and had measurable content learning gains, the ability to quickly develop their baseline knowledge of computing was overshadowed by complexities of the implementation details.

Over time the authors began using the teacher participants themselves as algorithmic components to illustrate fundamentals of computing. Each teacher, with a specific programmatic task, would act out that task, then the groups combined behavior, when organized correctly could solve problems given specific constraints. Limitations of this approach include both scale and re-use, and thus the need for a low-cost physical manipulative. The physical manipulative would need to perform some programmatic actions, be restructured, modified, removed, and finally intuitive enough to be used by all ages.

Thus an idea was born – what if you had a set of physical building blocks that could perform a predetermined yet modifiable action. When connected to other blocks, the actions of the set of blocks is determined by the structural topology (connections between the blocks). This mimics the structure and syntax of languages – including those we use to program hardware devices.

This basic idea was presented to a five-member capstone team as the two-page proposal seen in Appendix A. While assigned an engineering faculty advisor, the team also had the benefit of two additional advisors in the form an education faculty member as well as and industry sponsor. The three advisors worked collaboratively to guide the capstone team towards a viable solution while not interfering with their day-to-day interactions. While not involved in the daily challenges and roadblocks faced by the team, the advisory panel monitored the team’s progress using Github’s collaboration features (www.github.com/features), as well as through regular team meetings. During these meetings, the engineering advisor and project sponsor advisor would consult with the team on challenges providing guiding suggestions; the education expert would aid in providing additional context in order to ground the project requirements and objectives based on prior educational research results. After several meetings between the capstone team and advisory panel, the team put forward an official design proposal to design, implement and build the blocks of code system in a 3-month timeframe. That proposal formed the basis of their ultimate end product, a functional set of physical programming blocks. Ultimately, their solution won “Most Creative Senior Capstone Project” during an end of the year capstone presentation.

Technical Details of “A Block of Code”

The overall system design of “A Block of Code” consists of two primary components – blocks of user-selectable functions and a controller cape, which provided power and a global reset signal. This distinct separation of the two subsystems, shown in Figure 1, is what ultimately allowed the team to success at delivering a functional “A Block of Code” product. This separation also lowers the barrier of entry to use the blocks of code – the pre-programmed controller cape handles all of the extraneous setup and complexity of the system, allowing the end user to simply focus on manipulation of the physical blocks of codes.

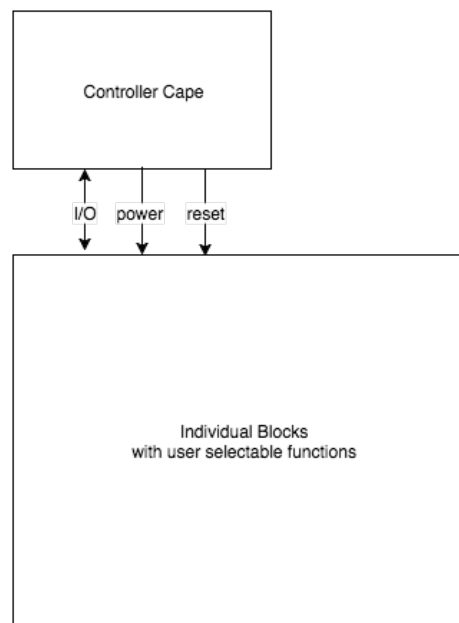


Figure 1: Level 0 - System Diagram for A Block of Code. The two primary subsystems allowed the team to work concurrently on three separate challenges.

As with most large run projects, a set of proof a concept “blocks of code” were designed and built prior to PCB fabrication. In order to exist the proof of concept stage, the team targeted three critical components of the system: a) intra-block communication, b) topology detection, and c) main processor token recognition. Because the team decided to abstract away control of the blocks to an external arbitrator, the team was able to work concurrently on the three major functions of the system, distributed across the two subsystems. The primary function of the block is to report its selected function in conjunction with the identity of its neighboring blocks back to the external arbitrator (or control cape). The control cape’s main functions are: network topology detection, block tokens parsing, program execution, and supplemental I/O and control of the blocks.

In order to create blocks of code that could communicate with each other in a reliable and intuitive way, the capstone team needed to devise a mechanism that would allow adjacent blocks to communicate. Given that a 3-dimensional space could allow for communication along six faces of a block, the design team and advisory panel decided that in order to maintain semblance of traditional programming constructs blocks would only communicate along four faces, retaining one of the remaining faces for user-based control and the final face as a base to rest on a solid surface (See Figure 2).

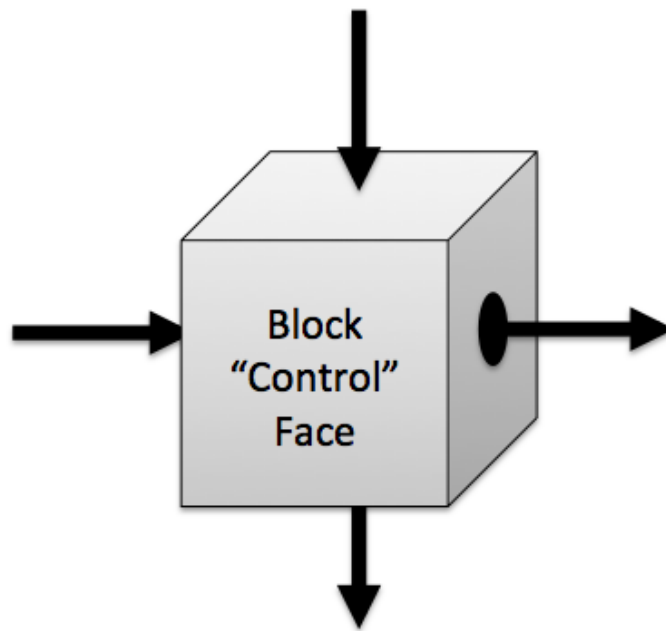


Figure 2: A Block of Code consists of five useable faces: four for communication and one for control.

To provide an infrastructure for intra-block identification and communication, each block contains a low cost AVR microcontroller, as well as four DB-9 connectors (two male and two female) positioned in such a way that the directionality of the collection of blocks is physically enforced. Each block is controlled by an ATtiny461 microcontroller, which is connected to power, ground, a 1M Ω potentiometer, a 2-neighbor local SPI-bus as well as a TWI-global bus. Figure 3 details the connections for power, clock and the 2-neighbor network.

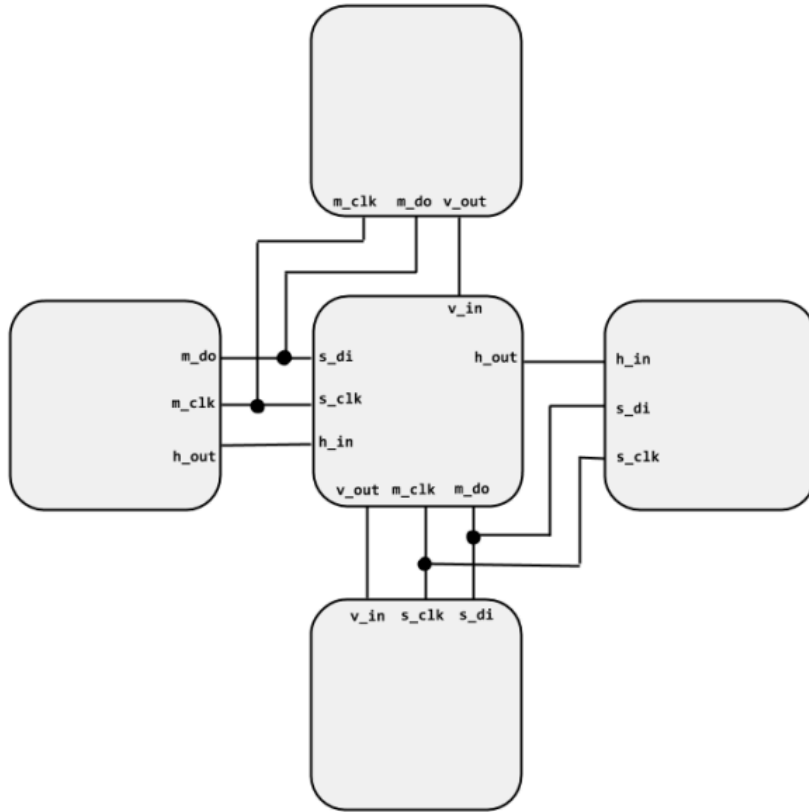


Figure 3: Power and distribution as well as 2-neighbor network (X & Y axis).

In order to discover the network topology of a set of blocks the control cape, in this case a BeagleBone, communicates along the local bus to a single block directly below it. A handshake protocol, which will be repeated for every pair of blocks, passes its own X and Y coordinates to the next block. Depending on where the handshake signal originates the receiving block will increment its own X or Y coordinate. The XY coordinate pair is then translated to a TWI address on the global bus. Topology discovery concludes when the handshake protocol yields no response – indicating there are no further blocks below or to the right of the ‘last block.’ The BeagleBone then queries each block using the TWI global bus and the XY coordinate derived addresses. The query returns a single byte payload seen in Figure 4. The information stored and transmitted by each individual block includes its neighbors, category type and current ADC value.

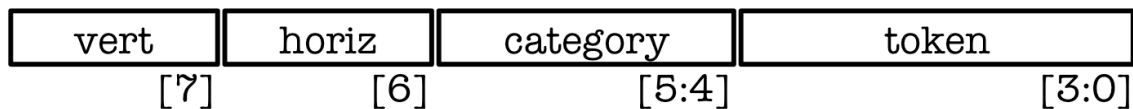


Figure 4: Single byte response from a block when queried. The 2 MSBs indicate presence of adjacent blocks vertically and horizontally. The next two MSBs contain the category of the block, while the 4 LSBs represent the value of the ADC.

The payload received from an individual block determines the functionality of a given block, ultimately used by the BeagleBone to construct a lexical mapping of the physical blocks. Using Table 1, the category and token (ADC) information is used to determine the exact functionality (lexical token) of

the user-tuned block. Users are able to tune the four types of blocks through use of a wheel attached to the potentiometer as seen in Figure 5. The four types of blocks include a value block, operator block, control block, and output block.

Table 1: Translation between ADC Token value and lexical token based on block category type.

ADC Token	Value Block	Operator Block	Control Block	Output Block
0	0	+	While	output1
1	1	+	While	output1
2	2	-	While	output1
3	3	/	End-While	output2
4	4	*	End-While	output2
5	5	^	If	output3
6	6	^	If	output3
7	7	%	else	output3
8	8	=	else	output4
9	9	≠	EndIf	output4
10	.	>	EndIf	output5
11	X	>	(output5
12	Y	<	(output5
13	Z	≥)	print
14	sum	≤)	print
15	count	!)	print

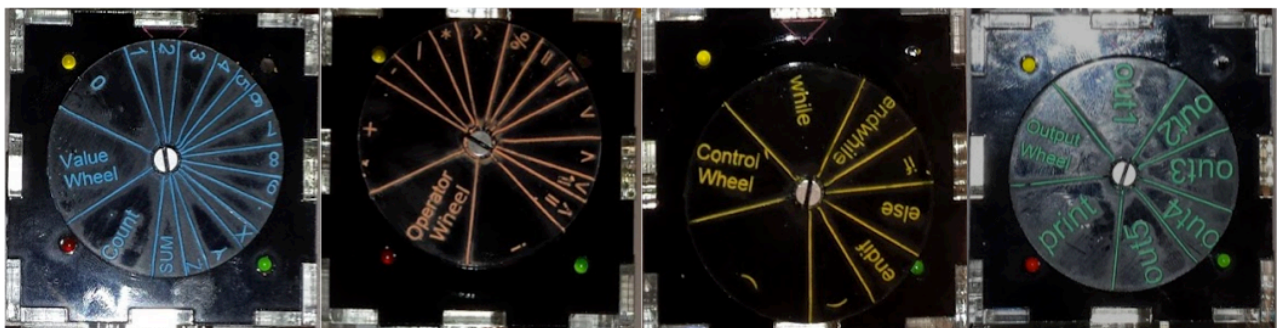


Figure 5: The four types of blocks with selections: Value (2), Operator (^), Control (while), Output (Out1).

Lexical analysis of the physical blocks occurs left to right from top to bottom. Figure 6 details the flow of the lexical analysis process. After lexical analysis, a parser and interpreter provide feedback to the blocks of code enabling a variety of Error and Status LEDs. In this implementation of ‘A Block of Code,’ the Error LEDs only provides an indicator that an issue exists, no other error message is provided to the end user. A syntactically correct program is executed, and outputs are logged both to the BeagleBone

standard output and passed through an output pipeline that drives the five customizable outputs listed in the Statement block column of Table 1. Figure 7 shows a sample program, and the flow through the orchestration script that runs on the controller cape with a buzzer attached to output 2 and a motor attached to output 4.

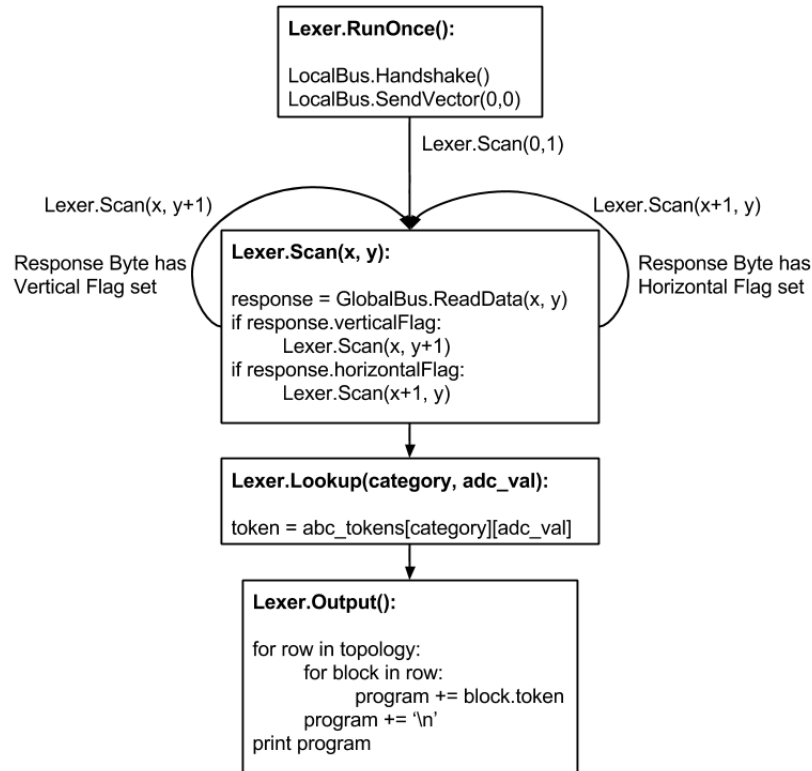


Figure 6: Lexer Flow Diagram highlighting the interaction between the lexing process and the block payload.

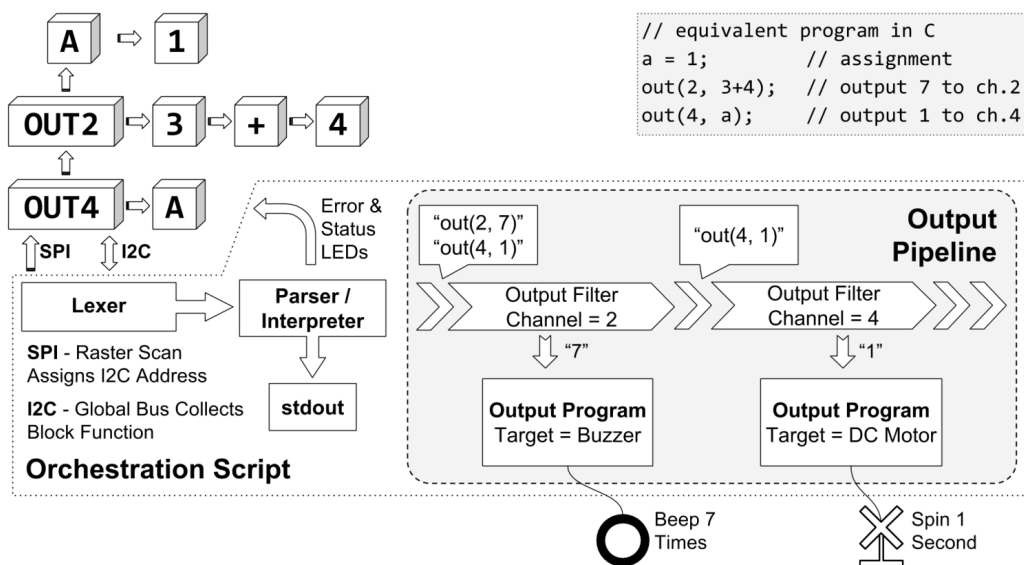


Figure 7: Interaction of the physical blocks and orchestration code running on the controller cape.

Results of Physical Implementation and Future Direction

The final physical product, a set of 50 “Blocks of Code,” were assembled, tested and then used in a hands-on activity. For the hands-on activity several K-5 students were randomly selected to interact with the blocks with minimal instruction. Of this small sample of students ($n < 10$), none were known to the researchers and none had programmed before. These students quickly engaged with the blocks asking questions and experimenting with the available structures and features. Since the event was co-located with an annual CS and ECE senior capstone presentation, many other capstone teams ventured to the activity and spent as much time as the K-5 students actively engaged in trying to solving a set of advanced algorithm – most competing to achieve solutions in the fewest number of blocks. A subset of pictures from the event is found in Figure 8.



Figure 8: K-16 students with "A Block of Code." K-5 far left / top center and older students right / bottom center.

Through observation and informal interviews, the authors were able to determine that the K-5 students, with no prior CS experiences, were able to construct simple programs to solve several algorithmic challenges through use of the “Blocks of Code.” While admittedly a small sample of K-5 students ($n < 10$) participated in the initial trial, the feedback from those students, faculty observers, and senior capstone peers on the concept of “Blocks of Code” was extremely positive. Use of manipulatives engaged novice and experts alike – with novices interested in just getting a working solution and experts trying to show off the most complex program in the fewest number of blocks. A sample guiding worksheet that was provided for older students during the activity is located in Appendix B.

The development of a set of physical set of “blocks of code” yielded a plethora of ideas and enhancements for future CS and ECE manipulatives. With respect to the existing concept, ideas for future

development include magnetic coupling for physical and communication coupling, user definable blocks, additional input sensor blocks, smaller form factor “puck-like” blocks, and dispersed mesh processing rather than centralized external processing. The idea of using foundational block structures to implement a complex system goes well beyond CS, and even electrical engineering, the block concept could easily be translated to biology, chemistry, and even traditional language studies. Finally, while the objective of “A Block of Code” was to create a physical manipulative, the authors also realize that this not always the most optimal delivery mechanism – virtual manipulation of 3D block forms may yield a reasonable compromise for spatial stimulation in working with abstract concepts such as those found in computer science.

The Study and Methods

This work looks at the impact of authentic value-added capstone projects on student’s soft skills by comparing results of a multi-year collaboration survey given to multiple senior capstone teams with data collected during use of an online code collaboration tool. The Senior Design Capstone team implementing “A Block of Code” was one of three separate capstone teams that were paired with an advisory panel that contained both authors as well as an additional engineering faculty member. For the purposes of this mixed-method study, the participants consist only of the student members of the capstone teams. Additionally, due to the uniformity of the group participants the only factor that will used to differentiate the participants are whether they were “A Block of Code” members (ABC) or not (BASELINE). For each participant, their communication and collaboration soft skill aptitudes as well as perceptions of their selected project were self-assessed immediately prior to and immediately following the six-month senior capstone project. This self-assessment consisted of both quantitative Likert-scale questions as well as qualitative open response questions. In addition, the advisory panel tracked several key statistics about any group communication or conflict – either within the entire group or between selected participants. Finally, the participants were all required to use GitHub (www.github.com) as their primary means of collaboration, communication initiation as well as record keeping. The aggregate data collected by GitHub of the students’ usage patterns, communication frequency as well as several other built-in metrics were used to determine relationships and communication efficiency and development during the course of the project. Ultimately, notes from end of project interviews were used to validate observed patterns between participant project perceptions, communication and soft skill growth and outward communication.

Study Results

When asked before and after the capstone project to gauge their interest in the project itself, the ABC team as a whole had a much more positive outlook than the baseline participants. Before and after, the ABC team scored their interest, on average, in the project at “Significant” (4.7 and 4.8 respectively on a 5.0 Likert-scale), while the other baseline participants showed decreases from between pre and post interest from “Moderately Significant” to “Moderately” (3.9 down to 3.2). Additionally, of all participants in the baseline capstone projects which were not specially targeted to students, only 10% showed an interest in continuing to work on their capstone project after graduation, this was in stark comparison to the ABC participants of which 80% showed an interest in continuing development post-graduation. The two groups, ABC and baseline, responses to “how and why did you deal with conflict?” were also quite telling both in their responses as well as the supporting evidence. The baseline participant’s responses focused around resolving conflicts “for the grade” and doing so through “establishing hierarchies and domain boundaries and contracts.” While this may model typical professional and industrial models, it does not promote the soft skills expected through ABET. The ABC team references solving problems through collaborative efforts, focusing on having subsystem champions that enabled discussions, brainstorming, moderation and support throughout their project, not for “the grade,” but rather to achieve a “self-imposed goal of a successful project.”

Figures 9 and 10 highlight the stark difference between the ABC and Baseline teams. Note that all of the teams were given projects designed by the same project sponsor, all teams went through the same selection process, and the same senior capstone experience, and the majority of each advisory panels consisted of the same members. The number of team and sponsor meeting was nearly identical. The only difference found between the ABC team and the Baseline was the participant interest in solving the problem for a purpose beyond that of the grade of the capstone project.

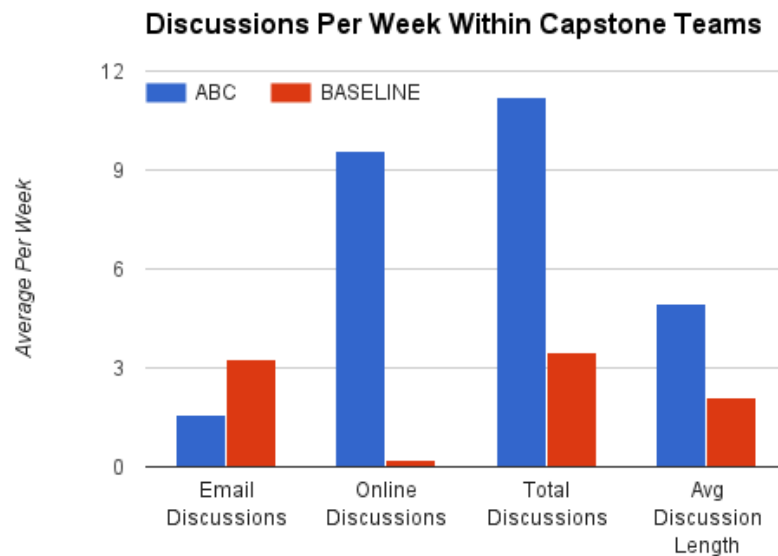


Figure 9: The Average Discussions and Discussion Lengths Per Week for the ABC team versus the Baseline teams.

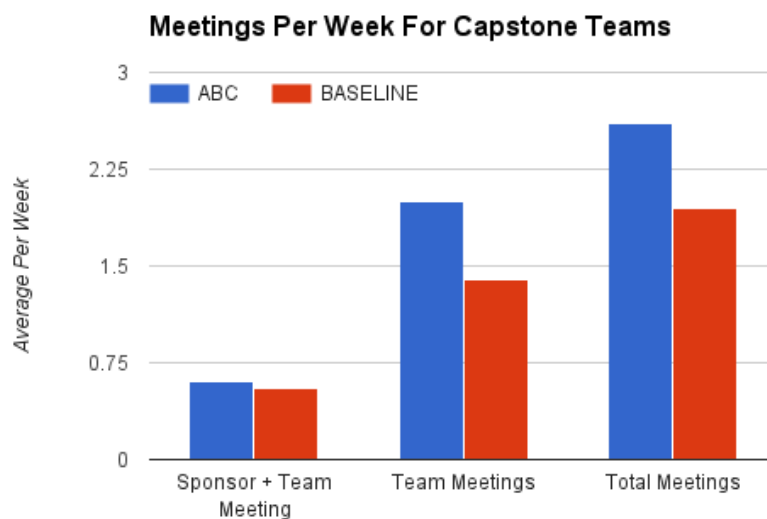


Figure 10: The Average Number of Meetings Per Week for the ABC team versus the Baseline teams.

Conclusion

The observed trends suggest that projects with community impact (irrespective of size or geographic constraint) foster increased communication, participation, and ultimately collaboration. The ABC participants self-identified as “caring about computing within an educational context.” While community impact is generally thought of a localized, geographically based concept, looking at professional, academic, and perhaps non-traditional communities can aid in the development of projects ideas for K-16 students. In the context of capstone projects, appealing to open needs of a community that the students recognize and are a part of can greatly influence their motivation and desire to succeed. In order to build a successful and diverse field of future ECE and CS students and professionals, we as a community need to develop pathways that go beyond traditional robotics and gaming-centric. We need to continue to develop interest in the field by exposing students of all ages to the real world applications and societal impacts of computing, and we need to build their confidence and foundational skills without requiring foundations in high-level abstract thinking skills to get started. We need to engage our students in the possibilities of computing not just in K-12, but into college as well.

Acknowledgements

The authors acknowledge the members of the “A Block of Code” capstone team: Nathan Bryant, Daniel Frister, Tyler Hart, Jacob Mickiewicz, and Greg Stromire; their ECE faculty advisor: Roy Kravitz; and, the Portland State University administration and faculty, for their continued partnership and development of high quality and innovative capstone products and experiences. Additionally, “Blocks of Code” and the other capstone projects referenced in this paper were funded by Erebus Labs, with funds used entirely for the physical components and fabrication of the end products.

References

- [1] Blackley, S., & Howell, J. (2015). A STEM Narrative: 15 Years in the Making. *Australian Journal of Teacher Education*, 40(7). <http://doi.org/10.14221/ajte.2015v40n7.8>
- [2] Nadelson, L., Callahan, J., Pyke, P., Hay, A., & Schrader, C. (2010). Teaching Inquiry Based Stem In The Elementary Grades Using Manipulatives: A Systemic Solution Report (pp. 15.1176.1–15.1176.18). Presented at the 2010 ASEE Annual Conference & Exposition.
- [3] Margolis, J., Fisher, A., & Miller, F. (2000). The Anatomy of Interest: Women in Undergraduate Computer Science. *Women's Studies Quarterly*, 28(1/2), 104–127.
- [4] Burns, B. A. and Hamm, E. M. (2011), A Comparison of Concrete and Virtual Manipulative Use in Third- and Fourth-Grade Mathematics. *School Science and Mathematics*, 111: 256–261. doi: 10.1111/j.1949-8594.2011.00086.x
- [5] Clements, D. H. (1999). “Concrete” manipulatives, concrete ideas. *Contemporary Issues in Early Childhood*, 1(1), 45–60.
- [6] Suydam, M. N. (1985). Research on instructional materials for mathematics. Columbus, OH: ERIC Clearinghouse for Science, Mathematics and Environmental Education.
- [7] Suydam, M. N., & Higgins, J. L. (1977). Activity-based learning in elementary school. Mathematics: recommendations from research. Columbus, OH: ERIC Center for Science, Mathematics and Environmental Education, College of Education, Ohio State University.
- [8] Gogtay, N., Giedd, J. N., Lusk, L., Hayashi, K. M., Greenstein, D., Vaituzis, A. C., ... Thompson, P. M. (2004). Dynamic mapping of human cortical development during childhood through early adulthood. *Proceedings of the National Academy of Sciences of the United States of America*, 101(21), 8174–8179. <http://doi.org/10.1073/pnas.0402680101>

- [9] Jensen, E. (2005). Teaching with the Brain in Mind, 2nd Edition. Association for Supervision and Curriculum Development.
- [10] Ottenbacher, K. J., Muller, L., Brandt, D., Heintzelman, A., Hojem, P., & Sharpe, P. (1987). The effectiveness of tactile stimulation as a form of early intervention. Journal of Developmental and Behavioral Pediatrics, 8(2).
- [11] Trowbridge, L. W., Bybee, R.W & Powell, J. C. (2000). Teaching secondary school science. Englewood Cliffs, NJ: Prentice Hall.
- [12] Ejiwale, James A. (2012). "Facilitating teaching and learning across STEM fields." Journal of STEM Education: Innovations and Research 13(3) 87.
- [13] Obama, B. (2016). "The President's Radio Address: Giving Every Student an Opportunity to Learn Through Computer Science For All." Office of the Press Secretary, The White House. Available <https://www.whitehouse.gov/the-press-office/2016/01/30/weekly-address-giving-every-student-opportunity-learn-through-computer>

Appendix A: ECE Capstone Proposal

A Block of Code:

Title: Physical manipulators for developing and implementing short programming routines

Project Short Name: A Block of Code (ABC)

Sponsors: BLINDED

Targeted Users: Infants to High School aged children, parents and their educators

Problem/Task:

Your task is to design the basic hardware / software structure for ultra-low cost, connectable “blocks” that mimic a simple-grammar programming language. The fundamental grammar should contain 7 types of blocks:

- numbers (ints/doubles) – stringing these blocks together forms longer numbers [0,1,2,3,4,5]; [6,7,8,9,.]
- variables
- assignment operators
- simple binary operators (*,/,+,%?):
- equivalence operators (<,>==, !=,<=,>=)
- control block start/end
- program start/end

While a majority of the logic for the entire operation could be housed in the program start/end or control block start/end – there are communication and connectivity challenges; **especially without exposing connectors.**

Background:

Algorithms, logical constructs and the foundations of programming are generally limited to computer-based instances. No physical manipulators exist that expose children to programming as they do for number and alphabet systems, mechanics, art, etc. Build a prototype for building blocks that can be attached through internal magnets, these magnets perform two actions 1) physical connect and 2) logical circuit connections.

Skill Requirements (Order of importance):

- System Design
- Ultra-small form factor design
- Ultra-low power/heat design
- Power and signal transmission through traditional wire & inductance

Other Details:

- IP Ownership: Joint/None: The goal is to release an open hardware device and software; joint ownership of the original design, publicly available for reuse under similar attribution
- Suggested Team Size: 4-6: EE, CE, CS tasks abound;
- Extension after End of Project: 50-100 parts run to distribute to high-needs schools as a mechanism to enhance/drive STEM as a college degree option.
- Publication Opportunities: Conference and Journal Publication both in K20 Education, IEEE Spectrum, and/or others (joint ownership: ECE team and Sponsors) [resume/CV builder!]
- No NDA required.

Requirements:

Must Have

- Hardware cost <\$3 per unit (average cost for a set 20 = \$60)
- Open Source Hardware Design & Board (can use “closed source” components: ASICs, uC, etc.)
 - Must Have a Multi-Chip solution – e.g. no single SoC;
 - Magnetic + Inductive coupling between blocks
 - Fundamental grammar functioning; assignment; binary operators;
- Open Software Repository (github)
- Power:
 - Ultra Low power operation
 - Built in power source / power pack
 - Separate “power” blocks?
- Set of blocks for each team member and two sponsors

Like to Have

- Hardware cost <\$2 per unit (average cost for a set 20 = \$40)
- Hardware:
 - Multi-function blocks – e.g. 6 (or 4) binary operators on a single block; depending on connections/which side “faces up” operation changes
 - Control Structure Blocks
- Set of blocks for each team member and two sponsors + 5 extra sets

Nice to Have

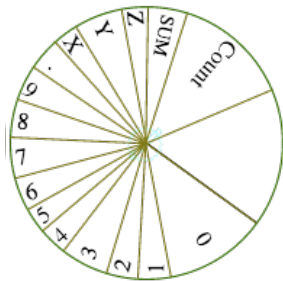
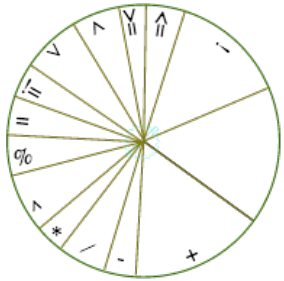
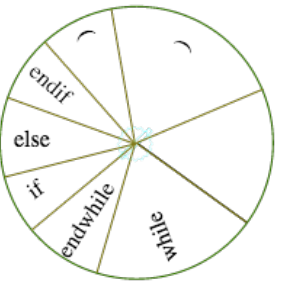
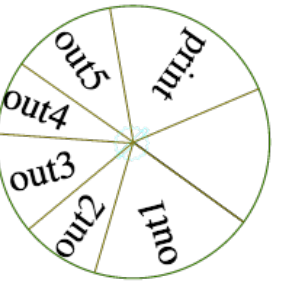
- Hardware cost <\$1 per unit (average cost for a set 20 = \$20)
- Power
 - Built in power source w/inductance recharge
- Computer GUI/Phone App: simulate a set of blocks (pre build) or take a picture of a set of blocks and execute code
- Set of blocks for each team member and two sponsors + 10 extra

Milestones:

- Hardware Evaluation to satisfy given constraints
- Main System: Modification of existing Open Hardware Design to create part or creation of custom PCB?
- Power System: Battery capacity + recharge?
- Communication mechanisms: Effective low cost communication and sync
- Implementation
- Development of full hardware solution
- Development of power system
- Development of communication system

Appendix B: Advanced Problem Section for Middle School to Collegiate Level Students

You have the following blocks available:

15 Value Blocks	15 Operator Blocks	15 Control Blocks	7 Output Blocks
			
<ul style="list-style-type: none"> • a variable {x, y, z, Sum, Count}, or • a value from 1-9 	<ul style="list-style-type: none"> • = is equality • checking NOT assignment 	<ul style="list-style-type: none"> • if, else & endif • while & endwhile • (&) 	<ul style="list-style-type: none"> • Sends output to specific channel
Sky Blue	Peach	Dandelion	Sea Green

Exercise #1: Convert & Print a temperature given in Fahrenheit into Celsius.

How many blocks does it take you?

Exercise #2: Write a program that computes Fibonacci numbers less than 25.

How many blocks does it take you? <50? <25?

[block area omitted for length]

Exercise #3: Create a program from scratch using as many blocks as you can.

[block area omitted for length]