

**Inspiring Curiosity through Computer Science Challenges:  
Students Modeling Chemistry using "Scratch" Visual Programming**

Lynne N. Cohen, Winter Park High School  
2100 Summerfield Rd, Winter Park, FL 32792

**Summary**

This six-week experience was a broad survey of the technology and specialties involved in developing the “Internet of Things” (IoT). IoT refers to the networks and interconnected objects which, through sensing environmental changes and communicating data, allow “smart” devices to work. These devices may automate tasks, improve health, conserve resources--or, some fear, may be used to manipulate human behavior. This program sought to educate a cohort of STEM teachers about the nature of IoT in order to promote the integration of related engineering and computer science concepts into middle school and high school classes.

In a world where communication is key, our role as educators includes not just written literacy but also digital literacy. The technological advances made in just the last twenty years have irreversibly changed our society; With sensors built into our smartphones, we can track our heart rate, exercise, and sleep. Many American households now welcome artificial intelligence into their home in the form of smart speakers. By understanding how these devices work--how sensors gather data, how networks pass and manage data, and how applications manipulate data--students will be able to participate in the conversations that will guide how we use our tech, and in turn, how our society as a whole operates. As educators, one accessible way to engage our students in this thinking is through incorporating coding projects into our curricula.

**Inspiring Curiosity through Computer Science Challenges:  
Students Modeling Chemistry using "Scratch" Visual Programming**

Lynne N. Cohen, Winter Park High School  
2100 Summerfield Rd, Winter Park, FL 32792

**Background**

IoT is most commonly understood among laypeople as home automation gadgets which communicate with our mobile phone applications via the cloud, allowing us to turn on and off lights, play music on “smart” speakers, control our thermostats, and view video footage of our homes from our phones. However, home automation is just one relatively minor application of IoT; in fact, IoT is a movement towards artificial intelligence and interconnectedness that will impact nearly every industry and will have far-reaching social implications as we struggle as a society with issues of privacy and security (Al-Fuqaha, Guizani, Mohammadi, Aledhari, & Ayyash, 2015). These issues are relevant to every member of society, not least among them public educators and those they are preparing for participation in a digitized society.

The RET: Comet program brought together professionals in three fields involved in furthering IoT technology: material/mechanical engineers, electrical engineers, and computer scientists. The research being done by these professionals impact many industries; in aerospace, cutting-edge composite materials are being used in hyper-sensitive sensors, strong and lightweight structural materials, and improved lightning strike protection (Gou, Tang, Lianga, Zhao, Firsich, & Fielding, 2010). In healthcare, facial recognition software may improve patient recognition and reduce malpractice (Jeon et. al., 2019), vital signs monitoring and behavior

monitoring may improve patient outcomes (Al-Fuqaha et. al., 2015), and responsive composites capable of flexing and sensing may improve laparoscopic procedures (Zhu et. al., 2017).

These IoT advancements are part of the digital landscape that sets the stage for our students as they exit high school, from the educational cradle into the expectations involved with earning a living. In 2001, Marc Prensky coined the term “Digital Natives” as a way of describing young people who have grown up accessing digital technology, and for whom digital objects are vital and omnipresent (Prensky, 2001). Almost 20 years later, the educational system is still stumbling to catch up with how to educate these “Digital Natives”. Some districts have gone “one-to-one,” pairing every student with a web-enabled laptop or device, but even in these progressive schools the use of technology can be clumsy as those who create curriculums struggle to translate traditional lessons into digital language--a language many teachers and administrators are not fluent in.

Yet our students are guaranteed to live as adults in a world of IoT, an increasingly neural-style network of millions of sensors and smart devices with which they will need to routinely interface (Al-Fuqaha, Guizani, Mohammadi, Aledhari, & Ayyash, 2015). In such a world, it will be vital for students to understand how digital objects work, and to be capable of procedurally and humanely solving problems as our society learns what it is to live symbiotically with artificial intelligence. Because of how broad IoT is, there is bountiful opportunity for students to personally connect to IoT concepts, either through interest in the disciplines working towards IoT design, or through interest on the many societal impacts.

Thus, educators now have an opportunity to tap into IoT in their classrooms in order to connect with and engage students while teaching meaningful lessons about digital interactions *in the context of* their own subject matter (Woo et. al., 2007). This is not to say that there is no room in the classroom for traditional lessons; after all, I've witnessed all ages experience joy and curiosity at the bubbling reaction between vinegar and baking soda. Yet, while preserving valuable traditions, we must make way for new lessons which will best meet the needs of today's students.

How are we to know what *are* the needs of today's students? It is difficult even for forward-thinking people to predict the skills that students will need ten or twenty years from now. Efforts such as the NGSS Engineering Design standards attempt to address this very issue by suffusing general engineering and problem-solving practices into science curricula around the nation (National Research Council, 2012). By developing and implementing lessons based on *posing problems* and *challenging students to design solutions*, educators prepare students with skills they can apply to nearly any industry or life path: "The person whose disposition is to be curious, confident, and eager to try something new, explore, engage, and try again learns what is needed to teach and lead regardless of age or position" (Gullen & Sheldon, 2014).

Science classes are obvious candidates for integrating engineering and computer science practice because they share similar challenges, mindsets, and problem-solving approaches. Engineering projects are increasingly common in science classrooms; however, many secondary science teachers still find it difficult to see the connection between their subject and *computer science*. This is an unfortunate blind spot, as it seems the only guarantee we *do* have about our

future is that people will be working with digitized objects (Moreno-León, Robles, & Román-González, 2016), and that is as true in an office setting as it is in a professional chemistry laboratory (Carr, 2014).

There are numerous options for exposing students to code through extracurricular activities (hour of code, coding clubs, science olympiad source code challenge), but unless a student specifically enrolls in a coding class--assuming that such a class is available in their school--they are unlikely to be exposed to coding in the first place. This is why, for the savvy science teacher, coding in the classroom can benefit the students while adding interest and variety to your classroom (Woo et. al., 2007). For reluctant teachers, there are five reasons to consider coding in your science classroom:

1. **Science courses teach problem-solving skills** grounded in practical applications & may directly translate to lucrative engineering & computer science careers (Carr, 2014).
2. **Computer science and laboratory sciences** share similar skill sets of defining problems, developing models, designing piece-wise solutions, interpreting data, and procedurally testing methods and solutions (National Research Council, 2012).
3. **Computing is required in modern experimentation** for modeling and calculation at the college level and beyond, i.e., MATLAB which uses logic and syntax similar to Java and other programming languages (Carr, 2014).
4. **Computer science challenges are authentic applications** for science modeling, formula use, and unit analysis – all essential to 6-12 chemistry, biology and physics standards.

5. **Students report increased motivation** for hands-on computer-based activities involving authentic problems, creative freedom, and development of student expertise (Gullen & Sheldon, 2014; Maloney, Peppler, Kafai, Resnick, & Rusk, 2008).

Perhaps the most important detail is that *it is not necessary for the teacher to be a coding expert*. The last point from the list above originates from a series of 2013 interviews with students on the topic of educational redesign and technology. Those interviewed expressed that the most engaging lessons involving technology were those that involved using web-based resources to answer questions and solve problems alongside their teacher. It was not a requirement for the teacher, in these cases, to be a digital expert themselves; students described favorite projects in which “students became the experts and the leaders” in developing skills with technology beyond the teacher’s expertise (Gullen & Sheldon, 2014). Similar feedback was found in a 2008 study of students in an urban after-school program, many of whom chose to spend their time with the Scratch programming language. In this study, students were self-motivated not only by the creative possibilities within the platform, but also by the emergence of local experts whose understanding of the coding practiced exceeded even the supervising adults (Maloney, Peppler, Kafai, Resnick, & Rusk, 2008).

That said, some understanding of code, current technology and research may help teachers scaffold lessons to support student learning. For this reason, experiences such as RET: Comet are valuable for teachers seeking to integrate engineering and computer science principles into their classroom. Following are details on research activities involved in this program.

**Research Activities: Overview**

Research activities were split into three two-week modules. In the first module, teachers were split into three sub-cohorts (4-5 each) to study under a professor who specialized in one of three sensor technologies: Dr. Jihua Gou with strain sensors, Dr. Hyoung Jin Cho with environmental sensors, or Dr. Reza Abdolvand with resonant MEMS sensors. For the first two weeks, I participated in Dr. Gou's strain sensor cohort. The strain sensor cohort covered properties of carbon nanotubes (CNT) and nanofiber (CNF), CNF composites, buckypaper, and carbon fiber plates. These cutting-edge materials allow advances in strain sensors and other products in medical and aerospace applications. Dr. Gou's module also touched upon additive manufacturing techniques (commonly known as 3D printing) which broadens the options for sensor manufacturing (Wang, Sparkman, & Gou, 2017).

In the second module, Dr. Mingjie Lin led an exploration of digital design including digital design including logic gates, binary, and hardware description language. He also shared with us the term deep learning, the concept of self-learning artificial intelligence that may change the way we interact with computers. Teacher trainer Jim Ebbert led the discussion on logic gates and logic tables, and graduate student Juan Escobedo introduced the Basys-3 FPGA board and the Verilog hardware description language. The module culminated in running a simple script in the FPGA board involving a progression of LED lights. Jim Ebbert, to support learning learning of coding principles, also introduced Java (with online compiler CodingBat) and Scratch visual block-based coding platform from MIT (Resnick et. al., 2009).

In the third and final module, Dr. Damla Turgut and graduate students Safa Bacanli and Ramin Izadpanah provided further practice with Java programming language, introduced network infrastructure and protocol, and provided Raspberry Pi motherboards to practice on. Also included in this module were command line, XML, and the Android SDK interface for mobile programming.

Further details of specific research experiences are recounted in detail below.

### **Module 1: Carbon Nanotubes and 3D Printing - materials & manufacturing advances for the development of better Strain Sensors**

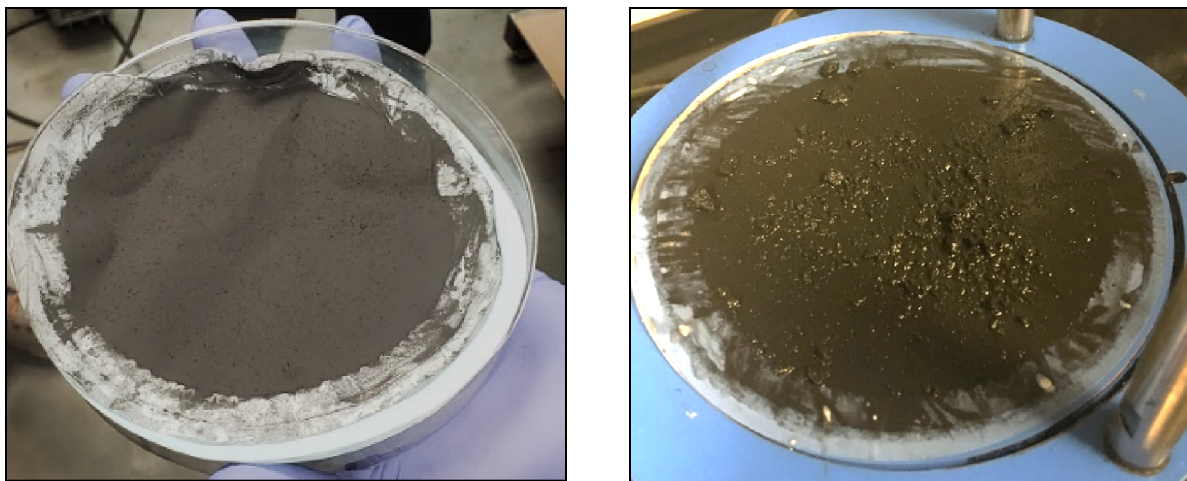
Dr. Gou's research includes the development of more advanced strain sensors using new composite materials and non-traditional manufacturing techniques. The properties of carbon nanotube (CNT) composites make them an excellent candidate for research on more sensitive and flexible sensors than what is widely available today. Carbon nanotubes, for example, are lighter and stronger than traditional building materials--including tensile strength, comparing against steel and aluminum (Wang, Liang, Wang, Zhang, & Kramer, 2004) .

CNTs also are incredibly conductive (thermal and electricity) depending on the structure and purity of the sample and the network structure of the tubes themselves, a network structure which can be interrupted through the introduction of stress--thereby changing their electrical resistance, and making them useful in sensing applications. In order to use CNTs in sensing applications, it is often common to make "buckypaper," a flexible paper-like compressions of CNT that can be further processed and combined with polymers to create a vast array of



materials which share the same properties as carbon nanotubes. These materials, as strong and flexible semiconductors, could lead to advances in flexible technology: wearable tech, more lifelike artificial intelligence, or surgical instruments that are less invasive and able to perform laparoscopic tasks with less damage (Zhu et al., 2017). CNTs are also a subject of research for sensitive aerospace applications. In one study, a CNT composite was formed into a plate and subjected to a simulated lightning strike to demonstrate lightning tolerance that correlated with conductivity of the material. This study demonstrated the potential of CNTs as a material for improved lightning protection for commercial airplanes and rockets (Gou et. al., 2010).

In lab, the process of creating buckypaper was demonstrated and practiced through sonic dispersal and pressure filtration. During this process, Disperbyk-190 surfactant and a “Q Sonica” Ultrasonic Processor were used to uniformly disperse the nanotubes in our solvent (DI water). A pressure filtration system removed the water and compressed the nanotubes uniformly (in theory) onto a piece of filter paper. The results of this lab varied from the demonstration day (6/3) to the group lab day (6/4). The result on 6/3 was a fairly uniform piece of buckypaper, while on 6/4 our mixture retained some aggregations of nanotube clusters. The procedure on both days was the same, except for the carbon nanofiber material used; on 6/3, we used Pyrograf PR-24-XT-HHT Carbon Nanofibers, while on 6/4 we used PR-24-XT-PS-OX. Results are shown below (see Figure 1).



*Figure 1.* Comparative results of buckypaper experiment, using two different carbon fiber materials. LEFT: using Pyrograf PR-24-KT-HHT carbon nanofibers, dried for 24 hours. RIGHT: using Pyrograf PR-24-XT-PS-OX carbon nanofibers (shown prior to drying).

In determining the reason for the difference of the two products, the cohort was led to research more on the origins of the two types of carbon nanofibers used. There are two main differences between the two materials: First, the “HHT” material is heat treated to remove additional impurities and to create a finer-grain product while “PS-OX” material is only mildly chemically treated to remove basic impurities (Pyrograf, n.d.). However, this material also has oxygen-containing functional groups added in, which should have theoretically made the material more polar and therefore more soluble in water. However, we did not find that to be the case, as the fibers aggregated rather than dispersing in the solvent. A possible reason for that is if the PS-OX material were older and had degraded or encountered pollutants that compromised its properties.

Despite the mixed results of the buckypaper experiment, it effectively demonstrated the processes used to create the materials and to form the buckypaper product, and the wide variations in quality possible with these materials.

Another manufacturing technique for carbon fiber composites is the creation of carbon fiber plates, involving resin impregnation and compression. The process involves layering “pre-preg” carbon fiber (woven sheets pre-impregnated with epoxy resin), and then using a vacuum seal and an autoclave machine to complete compression (see Figure 2). The carbon fiber plate that resulted from this process was incredibly stiff, dense, and difficult to break, much like those used in the manufacturing of certain sporting goods like carbon fiber bicycle frames or skateboards. As manufacturing technology improves for these carbon fiber plates, they can be used in new applications. In a 2015 patent, a design is described for a curved carbon fiber plate to be used in fracture repair in place of a traditional metal plate (Michel, 2015).



*Figure 2.* Laboratory procedures of layering pre-reg carbon fiber in an alternating pattern (LEFT) and sealing the product in preparation for vacuum and compression (RIGHT).

In week two, doctoral candidate Marie Lou Santiago presented her research on soft robotics using materials like carbon fibers and hydrogels (water absorbent polymer networks). One of the applications of these materials is in advancing laparoscopic techniques. At present time, laparoscopy is done with metal probes which, while less invasive than manipulation by hand, is still not perfect because the hard metal probe has a chance to damage the soft tissue it encounters. With flexible materials, opportunities for this type of damage is limited, and flexible probes would be able to reach locations and angles that a solid probe would not.

Some advancements in laparoscopic procedures currently are investigating how to create more degrees of freedom in their tools, such as one study where a tube is made more flexible through a segmented snake-like structure (Zhu et al., 2017), but Ms. Santiago's research could lead to materials which are inherently flexible. Some of the challenges she faces in her research, instead, is how to create necessary rigidness and grasping mechanisms, which is also part of the soft robotics investigations. Sensors are also a critical issue in soft robotics, and must be integrated for applications such as e-skins, sensors in organs, and fabric sensors.

After a review of stress (load/force/tension placed on sensor) and strain (geometric deformation of the sensor), it was understood that strain sensors work through linking an increase in resistance with an increase in strain (deformation). The sensitivity of a sensor is essentially how much resistance changes in response to strain. This is measured through "Gauge Factor" (change in resistance divided by change in strain), and a higher Gauge Factor (GF) means that the sensor is more sensitive to small changes in strain. Traditional strain sensors rely on conductive metal foil, which generally has a GF in the range of 2.0. In fact, their shifts in

Resistance can be so slight that a specialized resistance-sensing apparatus called a “Wheatstone Bridge” is often required to detect the changes (National Instruments, 2019); however, carbon nanofiber sensors have the capacity for a much higher GF.

In lab, the gauge factor of a carbon nanofiber sensor was calculated. The sensor was manufactured through inkjet printing and silicon lamination. Graduate student Jeydon Beyrooti provided samples he had created for his research, which were created in a manner similar to that described in a 2018 article (Li et al., 2018).

An MTS machine was used to apply axial stress (pulling stress) to the sensor. Deformation of the sensor was monitored on one computer with the MTS software, while another computer monitored the resistance data. The resistance data was collected with electrodes connected to a digital Wheatstone bridge, same as those used for traditional metal foil sensors.

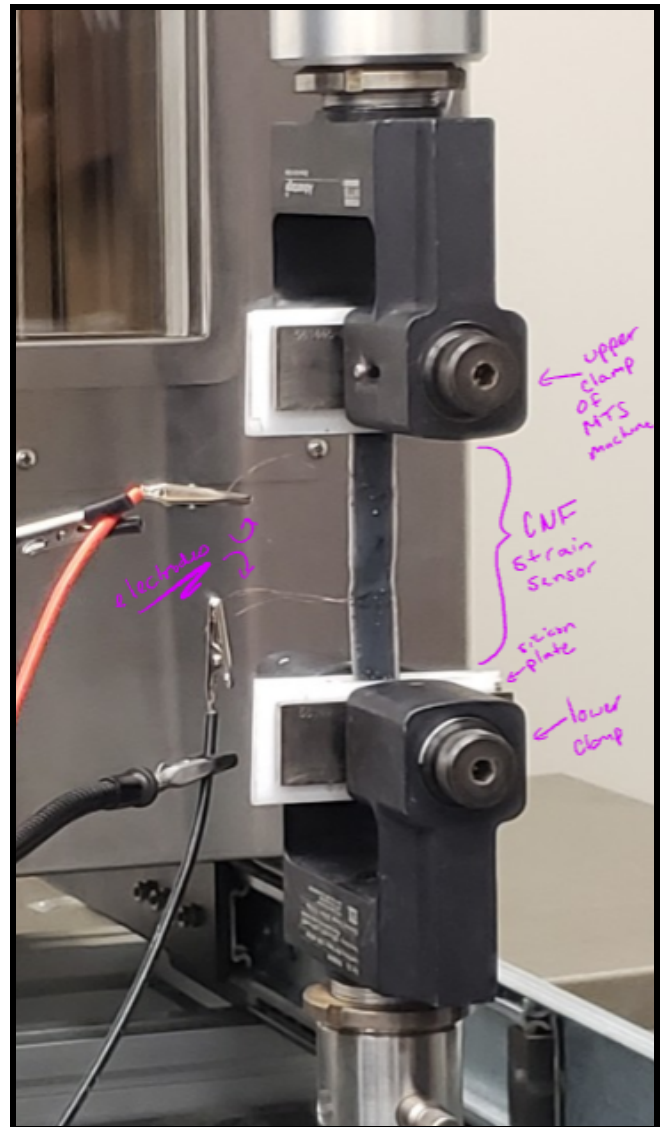


Figure 3. MTS machine testing of CNF-silicone composite strain sensor.

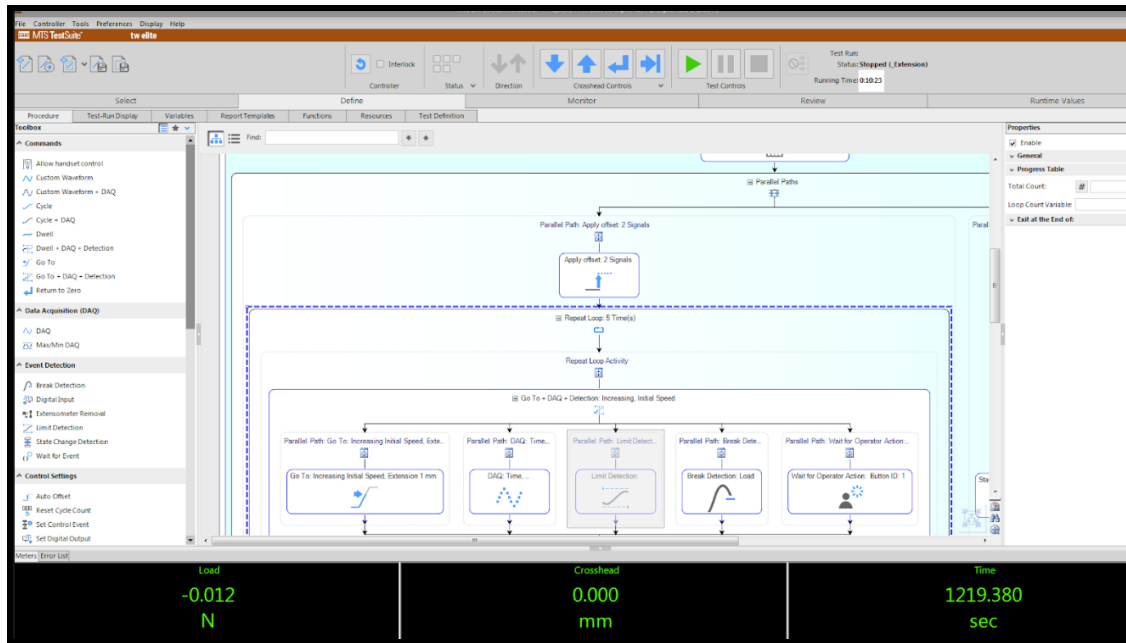


Figure 4. MTS TestSuite - Software for MTS Tensile testing machine

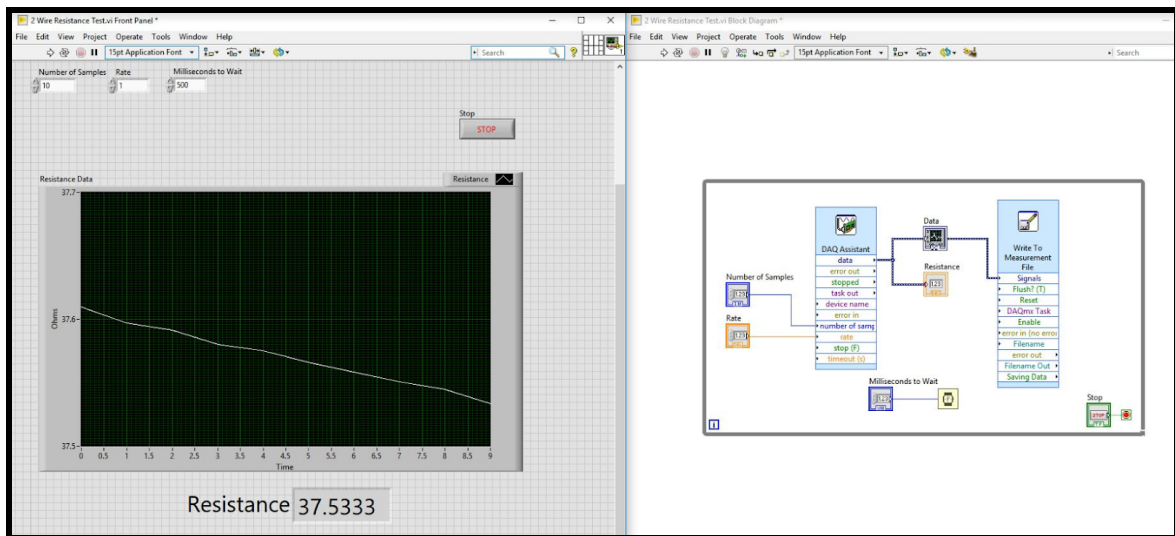


Figure 5. Software for digital Wheatstone bridge (resistance recording module)

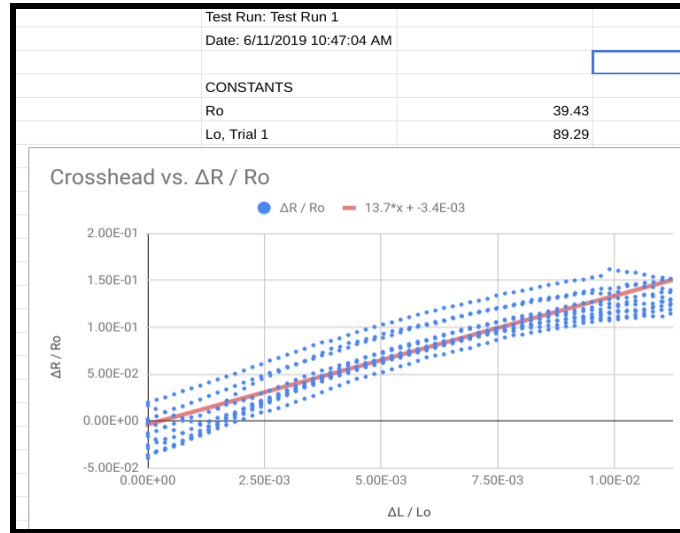


Figure 6. Graph of Results and Trendline equation

Gauge Factor :  $GF = \left( \frac{\frac{\Delta R}{R}}{\frac{\Delta L}{L}} = \frac{\Delta R}{R} \right) \quad GF = \frac{\Delta R}{R} \epsilon$

$GF = \frac{\Delta R}{R_0}$  →  $\Delta R = R_n - R_0 = 45.8 - 39.4 = \frac{6.4}{89.4} = 0.0716$   
 $\frac{\Delta L}{L_0}$  →  $\Delta L = L_n - L_0 = \frac{90.29 - 89.29}{89.29} = \frac{1}{89.29} = 0.0112$   
 $GF = \frac{0.0716}{0.0112} = \frac{14.5}{1}$

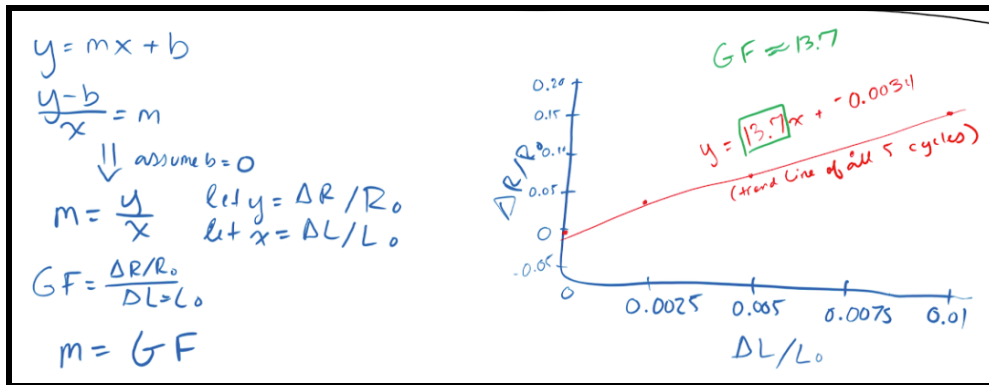


Figure 7. Calculations for Gauge Factor. Above: Algebraic calculations based on instantaneous data. Below: graphical analysis of data over the entire trial period, using trend line formula.

The data resulted in a gauge factor calculation of 13.7 for Jeydon Beyrooti's carbon nanofiber strain sensor, compared to a GF of 2 for a sensor made of metal foil. While the GF of this material is significantly better than metal sensors, there are challenges which mean the product is not ready for commercial availability yet. Other groups working on the same principles have published incredible results, manufacturing sensors with gauge factors in the hundreds and incredibly high stability, even after 10,000 cycles (Li et al. 2018). It's obvious that knowledge is advancing quickly in this area of research.

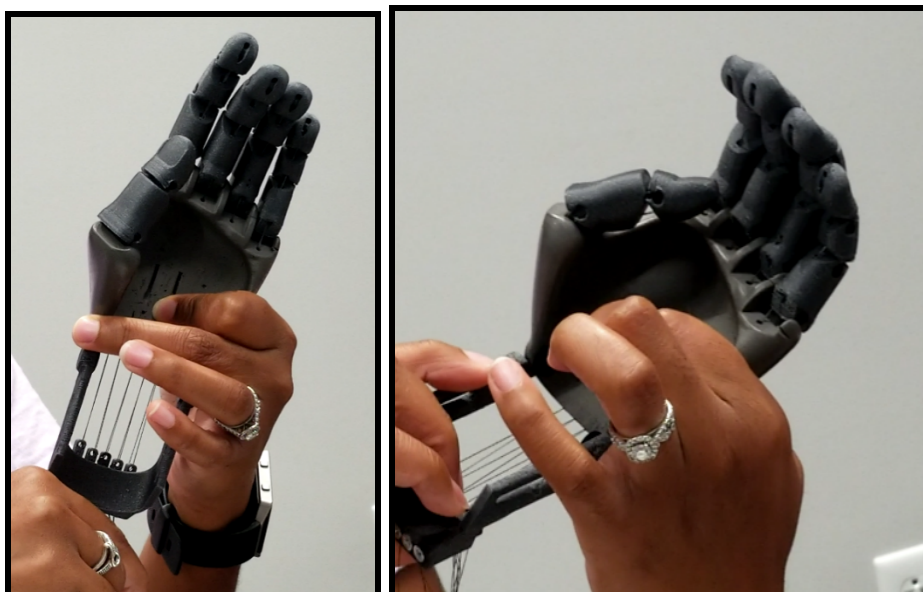
Over the two days during which we studied 3D printing, we learned about several types of 3D printers, materials used in printing, and applications of this technology. 3D printing, which is an additive manufacturing process (where many manufacturing processes today are subtractive in nature, similar to sculpting), is a quickly-growing area of research because of the variety of possibilities it opens. Essentially, all 3D printers work through depositing layers of solid material onto a substrate. The starting material may be a solid (such as polymers that are sold in spools of filament, melted and deposited), a liquid (such as resin which is selectively cured with UV radiation), or a powder that is cured or glued selectively.

There are several ways in which 3D printing and carbon nanotechnology intersect. The first way is the creation of carbon nanotube-infused composite materials; in Dr. Gao's lab, they have been able to suspend carbon nanotubes in ink to deposit on an inkjet printer. Theoretically, carbon nanotubes could be suspended in or mixed with a number of other materials and printed into any desired shape in order to create a sensor. Ideal solvents and composite materials are part of their study, though there is promising research in the area of combining CNTs with polylactic



acid, a common plastic known as PLA. A 2017 study comparing PLA composites of CNTs vs GPOs (graphene oxide sheets) shows further promise for CNT composite materials (Hu et al., 2017).

Another way in which 3D printing and carbon nanotechnology intersect is through biotechnology. On Thursday, graduate student Jonathan brought in a printed prosthetic hand, which can be manufactured using many commercially-available 3D printers and constructed using thread.



*Figure 8.* Prosthetic Hand manufactured with 3D printing, demonstrating biomedical applications of 3D printers and strong, light materials such as CNT composites.

The advancements in medical biotechnology that is possible using carbon nanofiber-based sensors includes the detection of both slight and broad muscular and joint movements, as well as vibrations associated with breath and heartbeat (Li et al. 2018). These

sensors could make possible much more sophisticated monitoring of a human under medical care, but could also be used for physical therapy, or used to store and relay data about human motion, improving the function and design of prosthetics. Embedding semiconductors such as carbon nanotubes *in* the production of prosthetics could allow them to transfer electrical impulses or heat as well, allowing--for the first time--to build “smart” prosthetics that behave and relay information more like an actual limb.

On the final day together, graduate student Shen demonstrated the process of growing carbon nanotubes via one of the most popular methods, CVD (Chemical Vapor Deposition). This procedure required a tube furnace, in which a catalyzed decomposition reaction takes place under extremely high temperatures. Toluene is used as a precursor, is mixed with an iron catalyst, then vaporized and pumped through the furnace carried by an inert gas, argon. As it decomposes with the help of the catalyst, the nanotubes grow within the tube furnace and we can extract it by inserting a silicone quartz substrate into the furnace on which the nanotubes can grow.

Prior to the lab experience, we sketched the lab setup and discussed how it would work. The sketch we developed is shown below, along with images from the lab, and the resulting nanotubes grown on the silica substrate during the lab.

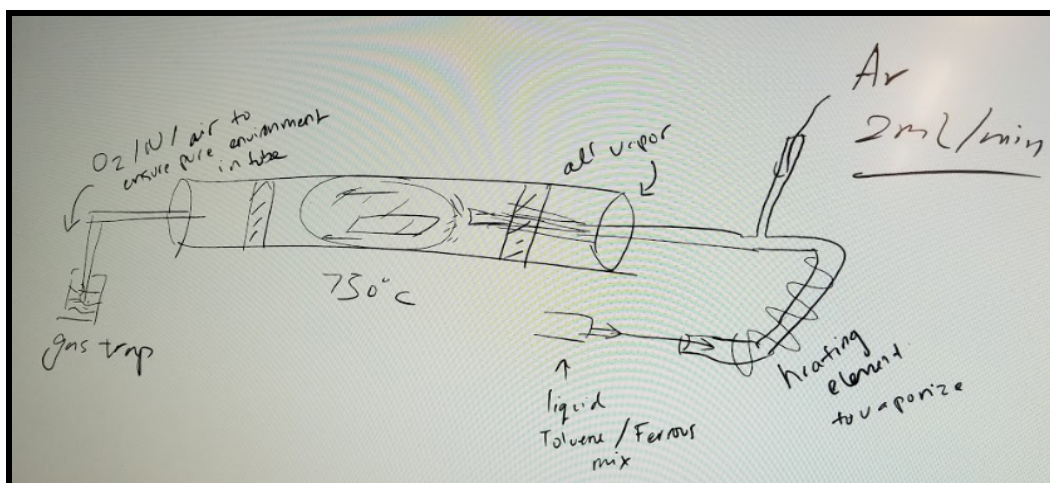


Figure 9. Experimental Design Sketch-up by L. Cohen



Figure 10. Above: Chemical Vapor Deposition Procedure: Toluene and Iron Catalyst (Left) and Tube Furnace (Right). Below: Resulting sample with deposited carbon nanotube layer.

## **Module 2: Digital Design - Hardware & Logic**

Electrical engineer Dr. Mingjie Lin introduced the cohort to the foundational concepts of digital design -- binary and logic gates -- and gave societal context for the importance of digital design. Digital design is central to the “Internet of Things” concept and is the foundation of what makes a cloud-based world possible. He addressed concepts such as Elon Musk’s “Starlink” satellites (which are being developed with the intention of spreading internet access to remote areas of the world), quantum computing (the advancement of computing which incorporates probabilistic behaviors and can, theoretically, handle more data and more readily model natural phenomena, including molecular behavior), and 5G technology (which would make it possible to run many more smart electronics on existing infrastructure due to the reduction in bandwidth competition). Dr. Lin left us with the question “What are the mathematics behind the IC technology? What design lies behind digital circuits?”. Our teacher trainer, Jim Ebbert, picked up from their and provided instruction for a majority of the following topics.

While the numbers we use in our daily lives are based in the “Base 10” number system, other number systems are valuable in computing and digital design. “Binary”, or the “Base 2” system, is valuable because the two values can represent “True” (1) and “False” (0). While it seems limiting to use only these two values, the permutations possible when constructing longer strings of these values quickly increase; in fact, they increase exponentially. While this is evident from basic statistical principles, the concept was effectively demonstrated by Mr. Ebbert through a discovery activity on the first day (see Figure 11).

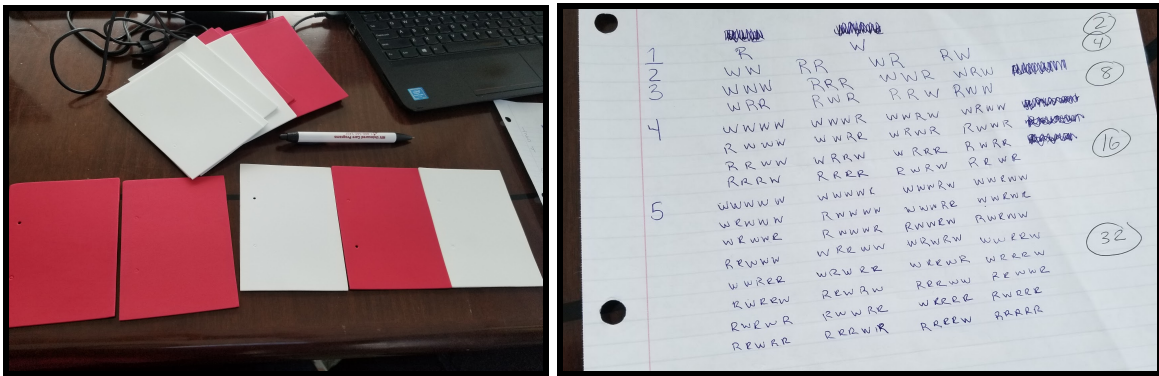


Figure 11. Discovery activity for Binary number permutations.

Further explorations covered conversion between base systems, determining negative values in binary, and performing addition operations on binary values.

Decimal Value	Place	Base 10 System			Base 2 System					Binary Value
		$10^2$	$10^1$	$10^0$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	
17	Value	=	1	7	1	0	0	0	1	10001
5		=		5			1	0	1	101
28		=	2	8	1	1	1	0	0	11100

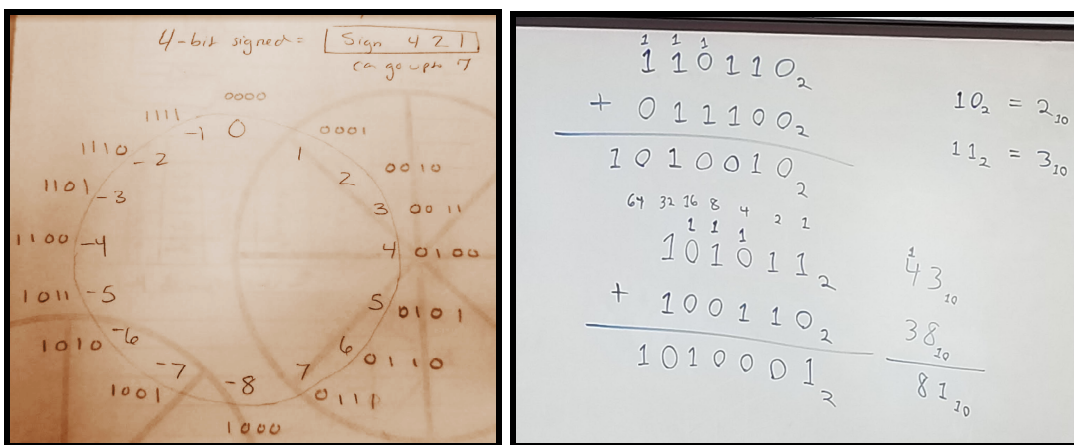


Figure 12. Exercises in binary value conversion, negation, and addition.

If binary is the foundational alphabet of computing, the language itself--its syntax--is boolean logic. Boolean logic is the passing of values of “true” or “false” based on logic “gates”, which are simple operators such as “and”, “or” and “not.” There are also secondary operators, but they are all combinations of these three basic logical concepts. “Truth tables” were used to rehearse the concepts. The truth tables essentially process each possible value of the inputs based on the logic operations desired. The logic operations can also be modeled with a circuit diagram using specific symbols.

After practice, our challenge was to create a series of logic gates that would solve for any addition problem; one series of logic gates had to result in every possible sum for two binary numbers (as well as any previous carry over digits), and another series had to result in the carry over digit for the next round. There were many solutions to this challenge; I attempted several, and came up with one involving seven logic gates, and two more involving six. Eventually, I was able to come up with a solution that involved only five logic gates, and that solution is shown in Figure 13.

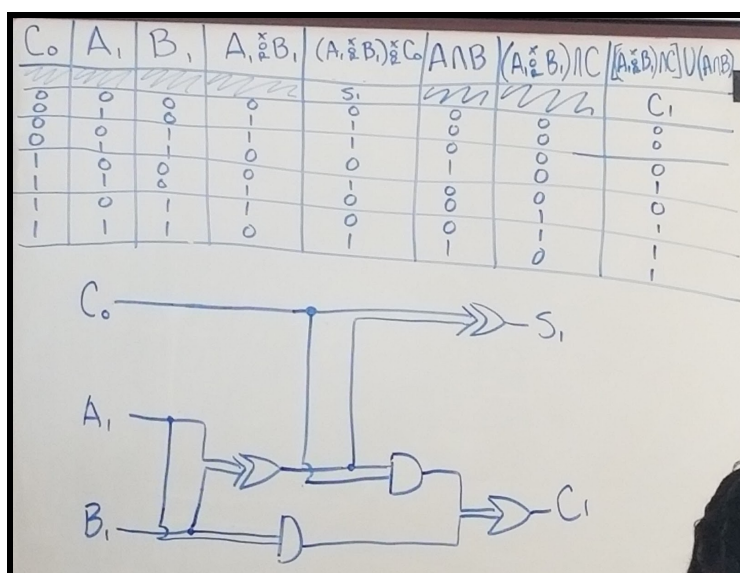


Figure 13. Five-gate solution to “Full Adder” Problem.

These exercises reflect the type of logic gates that comprise a Field Programmable Gate Array, which are comprised of a series of Lookup Tables (LUTs). LUTs can be repurposed freely on an FPGA chip through programming with a language like Verilog.

The module also included hexadecimal numbers (base 16), which uses the 10 digits we have societally designated (0, 1, 2, ..., 9) as well as an additional 5 digits designated by the letters A-F. Hexadecimal is useful in that a 4-bit hexadecimal value can be expressed by four 4-bit binary values, and so hexadecimal can be used as a shorthand for long binary strings:

Dec	up to $65535_{10}$				up to $4095_{10}$				up to $255_{10}$				up to $15_{10}$			
Hex	$16^3$				$16^2$				$16^1$				$16^0$			
	0, 1, 2, ..., D, E, F				0, 1, 2, ..., D, E, F				0, 1, 2, ..., D, E, F				0, 1, 2, ..., D, E, F			
Bin	$2^{15}$	$2^{14}$	$2^{13}$	$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1

Figure 14. Table Comparing Decimal, Hexadecimal and Binary Numbers

(Base 10, Base 16 and Base 2)

Hexadecimal, Binary, and Logic operations are all vital components in writing code, and are all components in Verilog (a hardware description language) as well as Java (a software programming language). Mr. Ebbert provided challenges of incremental difficulty on the CodingBat website (<https://codingbat.com/java>). Samples of problems are provided in Figure 15. *The content within the text box is my solution to the question.*

Write a method that returns 'true' if you can go, but returns 'false' otherwise. You can go if and only if you have a car and gas.

```
canGoUCF(false, false) → false
canGoUCF(false, true) → false
canGoUCF(true, false) → false
```

**Go** ...Save, Compile, Run (ctrl-enter)

```
public boolean canGoUCF(boolean haveCar, boolean haveGas) {
    return haveCar && haveGas;
}
```

Expected	Run		
canGoUCF(false, false) → false	false	OK	<span style="background-color: green; color: white;"> </span>
canGoUCF(false, true) → false	false	OK	<span style="background-color: green; color: white;"> </span>
canGoUCF(true, false) → false	false	OK	<span style="background-color: green; color: white;"> </span>
canGoUCF(true, true) → true	true	OK	<span style="background-color: green; color: white;"> </span>

**All Correct**

[next](#) | [chance](#)

Write a method that accepts three bits of a binary number as Boolean input parameters. Bit 0 is the least significant bit (the one's place). Bit 1 is the 2's place and bit 2 is the 4's place. Return an integer that has the value of that binary representation. So, for example, if bit2 is 'true'; bit1 is 'true' and bit0 is 'false' that represents the binary number 110. That number has a decimal value of 6, so you would return a 6. Look at the test data for additional examples. Please only use ONE return statement! (This means you will need to make a variable to temporarily store the results).

```
fromBinaryUCF(false, false, false) → 0
fromBinaryUCF(false, false, true) → 1
fromBinaryUCF(false, true, false) → 2
```

**Go** ...Save, Compile, Run (ctrl-enter)

```
public int fromBinaryUCF(boolean bit2, boolean bit1, boolean bit0) {
    int binaryNum = 0;
    if (bit0) { binaryNum = binaryNum+1; }
    if (bit1) { binaryNum = binaryNum+2; }
    if (bit2) { binaryNum = binaryNum+4; }
    return binaryNum;
}
```

Expected	Run		
fromBinaryUCF(false, false, false) → 0	0	OK	<span style="background-color: green; color: white;"> </span>
fromBinaryUCF(false, false, true) → 1	1	OK	<span style="background-color: green; color: white;"> </span>
fromBinaryUCF(false, true, false) → 2	2	OK	<span style="background-color: green; color: white;"> </span>
fromBinaryUCF(false, true, true) → 3	3	OK	<span style="background-color: green; color: white;"> </span>
fromBinaryUCF(true, false, false) → 4	4	OK	<span style="background-color: green; color: white;"> </span>
fromBinaryUCF(true, false, true) → 5	5	OK	<span style="background-color: green; color: white;"> </span>
other tests		OK	<span style="background-color: green; color: white;"> </span>

**All Correct**

Figure 15. Codingbat Problem Samples - Problems written by J. Ebbert (n.d.)

Building upon the conceptual foundations of logic tables and coding syntax, graduate student Juan Escobedo introduced the field programmable gate array (FPGA). A major issue in processor development is efficiency - how to use your hardware limitations to get out the most computing power. The challenge, then, is how to simplify your functions. On a hard-coded processor, we can only simplify functions through simplifying logic gates with constructs such as the Shannon expansion theorem (which uses identities to simplify an expanded formula), or the Karnaugh Map (where one matrixes the results of a function and writes a shorter function that achieves the same result) (Source: Lecture).

On an FPGA, the solution to this problem is perhaps one of the most straightforward, which is to create a “look-up table” (LUT). In this, a function fills in the results of a table, but the



lookup list *only* stores the result values, not the function that was used to fill it in. The results are ordered in ascending order of binary values, so the LUT does not even really require the inputs - the results can be fetched from their position in the table. In this way, the processor can be changed to perform different functions by simply changing the values in the LUT. Through the use of LUTs, the FPGA is essentially able to mimic the behavior of a processor without hard-coding the functions and logic gates that are normally integrated into a processor chip.

Other essential FPGA concepts are “flip-flops” and “latches”. These small functions process, temporarily store and pass information, and play a role much like RAM. There are two main types: the “D flip-flop” temporarily stores information and passes it on as it was received. It involves a clock to time the increments in which to store the data; Every so many ms, it stores the information given to it and passes it on. This is useful in that it is how RAM works, storing data temporarily so that it can be used and passed on. “Set-reset latches” act like a switch, turning a pathway off or on; it has the ability to pass information on as-is like a D flip-flop, or to reset.

These concepts were demonstrated using the Basys-3 FPGA board, which is an FPGA specifically designed for introductory users to train on (Digilent, n.d.). The Basys-3 itself is comprised of a number of physical features, though lecture focused largely on the 16 switches near the bottom of the board (see Figure 16), and the corresponding lights. Each of these is programmed as an array.

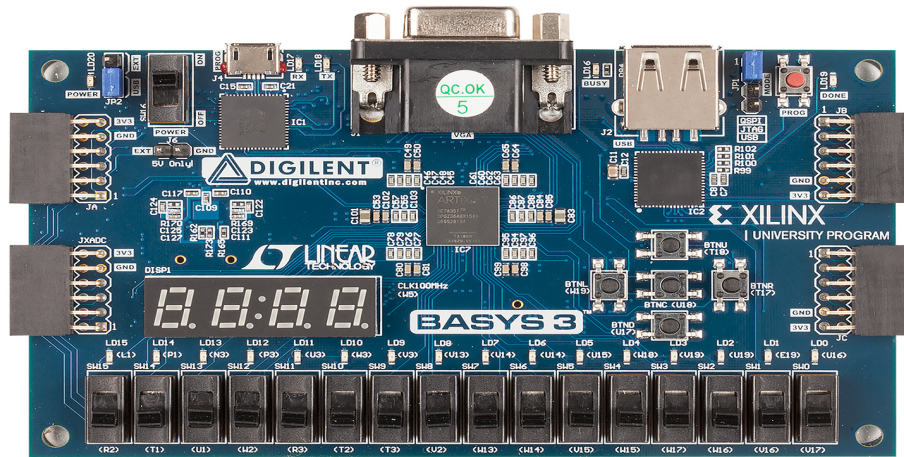


Figure 16. Basys-3 FPGA Board (Digilent, n.d.)

Two files are required to run a program on the Basys-3. The first is an “XDC File”, a pre-written file from the manufacturer that you edit to define different board features as variables or outputs they you will call in your logic file. The file is freely available on [github.com](https://github.com). The second required file is the “logic file,” which contains your actual program. The logic file is edited through a program called “Vivado” and is written in a language called “Verilog,” which is under a category called “Hardware Description Language” (HDL).

There are two main variable types in Verilog. Simple logic functions can be used with the type called “wire” while flip flops & clocks use the term “register.” “Wire” is literally data being passed along a wire, and is temporary. “Register” stores the data in a form of memory. However, it’s different than regular memory in that it is more limited, and exists on the chip (instead of on a separate physical structure).

Operator	Vivado	Java
AND	&	&&
OR		
NOT	~	!

Figure 17. Core Logic Operations and the Symbols used in Vivado (Hardware Description Language) vs. Java (Programming Language)

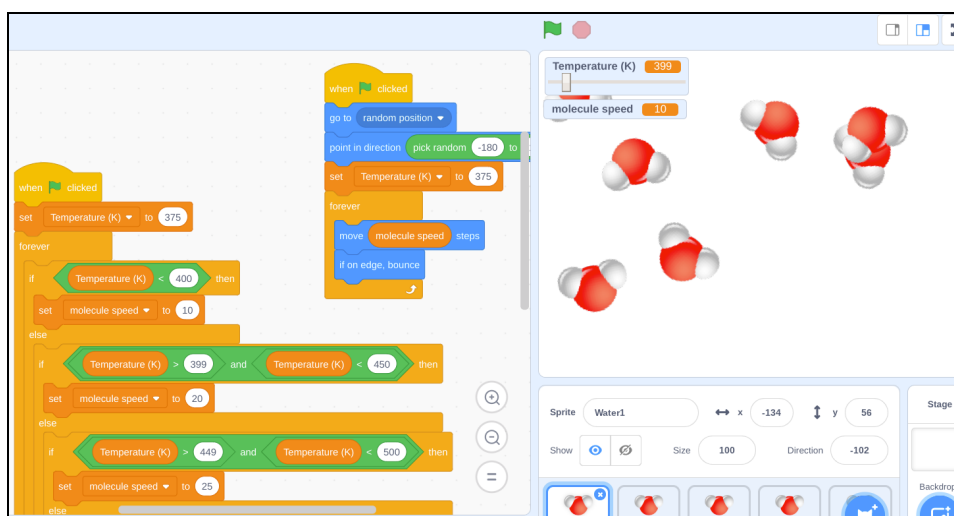
Verilog Code Samples	
Define an “AND” function:	Define an “OR” function:
<pre>public int sum (~) { } module and(in1, in2, out)     wire in1, in2, out;     assign out = in1 &amp; in2; end module</pre>	<pre>public int sum (~) { } module or(in1, in2, out)     wire in1, in2, out;     assign out = in1   in2; end module</pre>
Both at Once:	
<pre>module andor(and1, and2, and_out, or1, or2, or_out)     input and1, and2, or1, or2;     outputs and_out, or_out;     wire and1, and2, and_out, or1, or2, or_out;     assign and_out = and1 &amp; and2;     assign or_out = or1   or2; end module</pre>	

Figure 18. Vivado code samples provided by J. Escobedo

Hands-on verilog syntax practice was provided using an online compiler at

[www.tutorialspoint.com/compile\\_verilog\\_online.php](http://www.tutorialspoint.com/compile_verilog_online.php).

In this module, we also were provided free time to practice with various online compilers and coding websites such as the “Scratch” program available at <http://www.scratch.mit.edu>. I created models that could be used in my Chemistry course and considered how to introduce my students to the freeware, encourage imaginative play within the resource, and implementing engineering-based lessons based on using the resource to create useful models for approaching chemistry problems.



1. A limited function calculator for mole conversion problems:  
<https://scratch.mit.edu/projects/318420521/>
2. Visual models of [solids](#), [liquids](#) & [gases](#) using Kinetic Molecular Theory.
3. A model of a helium atom: <https://scratch.mit.edu/projects/318444659/>
4. A simulation of electron excitation & return to ground state:  
<https://scratch.mit.edu/projects/318603642/>

Figure 19. Sample Scratch Projects, Authored by L. Cohen

I developed these projects in order to acquaint myself with the Scratch platform, but also to use in my class both as models and as inspiration for student projects. I intend to leave these projects as they are, relatively rough and with superficial interaction, as I would like to give students a taste of what is possible while allowing them to independently imagine improvements and to make their own versions of these or other projects.

### **Module 3: Java, Networking Infrastructure, Raspberry Pi, Android Programming & XML**

The module began with a reading of the paper “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications” (Al-Fuqaha, Guizani, Mohammadi, Aledhari, & Ayyash, 2015). We discussed that the major goal of IoT is ubiquity--the wide availability of smart devices which communicate seamlessly with each other and communicates useful information with humans--and a major challenge in the way of reaching this goal is the existence (or lack thereof) of universally accepted protocols.

It is not sufficient to suggest just one protocol, as all smart devices operate in a layered infrastructure, and each layer has its own purpose and needs. For example, the protocols needed for the internal hardware--the design, internal hardware language, and any GUI that may exist on it--will obviously have different requirements than the language used to communicate with the cloud database to which the object connects. There are generally five layers which must be considered, including: The “object layer” (all sensors and actuators, which collect environmental information); the “object abstraction layer” (in which the sensors are addressed and given virtual identities, and initially begin passing digital information); the “service management / middleware layer” (pairing application data with requester based on locations and names); the “application

layer” (providing the services requested by the customer and providing a user interface); and the “business layer” (supporting the other layers with economic infrastructure and oversight, including design, analysis, and monitoring of IoT system elements, as well as R&D for future designs).

There are generally six required elements in IoT technology, several of which may need to be addressed at any given layer. The six elements of IoT are as follows:

- **Identification** - Unique object identifiers & Addressing within a network.
- **Sensing** - Gathering data & Communication it to a “data warehouse, database, or cloud”
- **Communication** - The ability to pass information over noisy and/or lossy links.
- **Computation** - Processing units & Applications that act as the “brain” of IoT.
- **Services** - Generally classified as one of these four:
  - Identity-Related service, Information Aggregation, Collaborative-Aware, and **ubiquitous services such as “Smart buildings”** - use “Building automated services” to regulate & optimize environment, safety, etc.; “Intelligent Transportation Systems” including but not limited to smart cars and smart roadside equipment; “Industrial automation” for manufacturing without human intervention; “Smart grids” for more efficient energy use
- **Semantics** - A universal language protocol that allows all objects to communicate information to any database, to be used for any application. Ex: XML

In the table to the right, each of these elements is shown with some of the most popular protocols/languages for that category of task. In the following section, the authors go on to discuss common standards currently being used in IoT applications & objects. They discuss application protocols, service discovery protocols, and infrastructure protocols, in the hopes of highlighting benefits and challenges of each one with the goal of working towards protocol ubiquity. At the current time, many categories do not have a clear winner; therefore, there is room to improve, consolidate and innovate in order to move towards IoT ubiquity (Al-Fuqaha et al., 2015).

Lastly, the authors discuss challenges in IoT such as availability, reliability, mobility, performance, scalability, interoperability, security, management, and trust. These are more logistical and infrastructure challenges of IoT, some of which are related to protocol barriers, and some of which are separate issues. The authors suggest some methods of overcoming these barriers (such as infrastructure changes to support mobility and reliability), though others remain (like security, which is a major barrier for consumer adoption) (Al-Fuqaha et al., 2015).

TABLE II  
BUILDING BLOCKS AND TECHNOLOGIES OF THE IOT

IoT Elements		Samples
Identification	Naming	EPC, uCode
	Addressing	IPv4, IPv6
Sensing		Smart Sensors, Wearable sensing devices, Embedded sensors, Actuators, RFID tag
Communication		RFID, NFC, UWB, Bluetooth, BLE, IEEE 802.15.4, Z-Wave, WiFi, Wi-FiDirect, , LTE-A
Computation	Hardware	SmartThings, Arduino, Phidgets, Intel Galileo, Raspberry Pi, Gadgeteer, BeagleBone, Cubieboard, Smart Phones
	Software	OS (Contiki, TinyOS, LiteOS, Riot OS, Android); Cloud (Nimbits, Hadoop, etc.)
Service		Identity-related (shipping), Information Aggregation (smart grid), Collaborative-Aware (smart home), Ubiquitous (smart city)
Semantic		RDF, OWL, EXI

Figure 20. Protocols (Al-Fuqaha et al., 2015).

An entire section is devoted to the discussion of “fog computing” which is essentially the delocalization of cloud computing; Whereas the early days of cloud computing held most of the computing and processing power in a centralized location in a server farm, Fog computing relocates some computing tasks to more regional and localized locations while maintaining connection with the cloud. A portion of this infrastructure would be made possible through improving better horizontal communication, meaning communication among IoT devices. IoT will not be fully realized until this horizontal communication is seamless and the data collected from these variety of sensors is collected and presented in a useful way.

Lecture covered the origin of the Java programming language, which was developed to be a simpler language than C++, giving up some lower-level uses of C++ but allowing us to write web applications, and for more people to access programming.

Java Applications have layers which allow it to work; on top of the Operating system, within the Java native Interface, there is a Java “virtual machine” which includes a class library and common language. This allows any program written in Java to run on any hardware as long as the Java virtual machine is present. The JVM acts as a “runtime environment” and runs any Java .class files within SUN’s specifications (SUN is the originator of the Java language).

The *source code* is simply the instructions written in a programming language. A high-level language such as Java requires a *compiler* in order to translate instructions to the machine it is running on. A compiler stores the code beforehand as an executable file, therefore it runs quite quickly. Another option is an “interpreter”, which does not store the source code,



and simply translates into machine language at runtime--therefore, this is more flexible, but slower on execution.

An IDE Is an “Integrated Development Environment” which is a compiler that includes an editor & tools such as a syntax checker & debugging to allow you to more easily write in Java. In the RET program, we were asked to download the Java compiler and IDE “Eclipse”, though many of us faced challenges downloading and/or using Eclipse, and so instead relied on BlueJ or compileonline.com, which worked just as well for practicing Java coding.

**The steps from defining a problem to building a software solution is:**

1. Requirements analysis
2. Design
3. Implementation

*Testing/Debugging is a separate phase, but is also incorporated at other stage.*

A Java program is composed of building blocks of objects, methods, and classes. A *class* is where all the code in a Java program resides. In Java, a *program* is actually called a “**class**”. A class name is generally written with a capitalized first letter such as “Classname”. The filename and the class name must be the same! A class specifies the attributes that objects within it can have (what kind of data they are) and provides methods-- or actions that objects of that class can take.

A “**method**” is a set of instructions within a class meant for objects of that class.

```
public class ClassName {  
    public static void main (String []args) {  
        for (int i=1; i<=10; i++) {  
            System.out.println("(1,-1), ");  
        } } }
```

*for the method “main” above:*

“static” means it applies to the whole class, not a specific object within the class.

“void” means there is no return statement - not meant to return a particular value

In order to write Java code, a few things must be taken into consideration: First is the Algorithm (in pseudo-code), including sequence, decision, repetition, input, output, assignment. Second is Data & Memory requirements, covered by data type, variables, constants, initial values, etc. Finally, comments and methods. **Comments** are written either as single lines begun by // , or as multiple lines enclosed in /\* and \*/. **Variables** or **Identifiers** are user-defined and can be defined as one of the following data types: **int** (numeric integer), **double** (for decimal characters), **char** (for any single character including letter, number, or symbol), **boolean** (true/false), and **String** (any sequence of zero or more characters).

Declaring a “**constant**” is similar to declaring a variable, except that you begin the declaration with the word “final”, and constants are generally named in ALLCAPS.

```
final type name = value  
final int SPEEDOFLIGHT = 300;
```

Writing to **output** (where the “output” is a literal value, a named variable or constant, or an expression):

```
System.out.println( output );      //prints the output and then returns to the next line.
System.out.print( output );        //prints the output and then keeps cursor on same line.
```

Assigning a variable is done as we have been doing in previous weeks:

```
resultVariable = expression;
```

Evaluations are done with an order of operations: Bracketed sub-expressions first; Operator Precedence (PEMDAS); Left to Right. *Remember that division will always be integer division unless one of the values has a decimal in it, regardless of if you are using a variable that has been previously defined as a double.*

Mutually exclusive IF options:

```
if ( x < 0 ) {
    System.out.println( “Negative”);
} else if ( x == 0 ) {
    System.out.println( “Zero”);
} else if ( x < 5) {
    System.out.println( “1 – 4 inclusive”);
} else {
    System.out.println( “>= 5”);      }
```

Options for repetition statements:

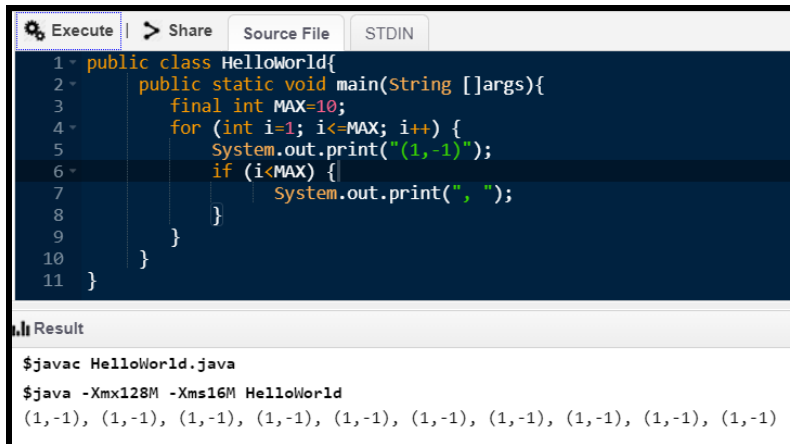
<code>while (condition) { statement; }</code>	<code>\\ Repeat 0 or more times</code>
<code>do {statement } while (condition);</code>	<code>\\ Repeat 1 or more times</code>
<code>for ( init; condition; update) { statement; }</code>	

We also discussed altering code to reduce redundancy & increase efficiency of code. We learned to recognize “infinite loops” and attempt to avoid these pitfalls. We also learned some other common pitfalls, like errors resulting from dividing by zero.

<i>Avoid “divide-by-0” error</i>	<pre> if ( y = 0 ) { System.out.println( “Error: can’t divide by zero”); } else {      z = x / y;System.out.println( “The result is “ + z );      } </pre>
----------------------------------	--

**Standard Classes** are pre-written and provided by the Java Development Kit, that can be called in order to perform tasks like reading user input. *Scanner* is an example of a “Standard Class” (see the Hands on Practice below). We touched upon **Wrapper Classes** which can manipulate a value or change its type; however, we did not go into great detail on nor practice with these classes.

Hands-on practice was done using compileOnline ([https://www.tutorialspoint.com/compile\\_java\\_online.php](https://www.tutorialspoint.com/compile_java_online.php)) (See Figure 21)



```

1 public class HelloWorld{
2     public static void main(String []args){
3         final int MAX=10;
4         for (int i=1; i<=MAX; i++) {
5             System.out.print("(1,-1)");
6             if (i<MAX) {
7                 System.out.print(", ");
8             }
9         }
10    }
11 }

```

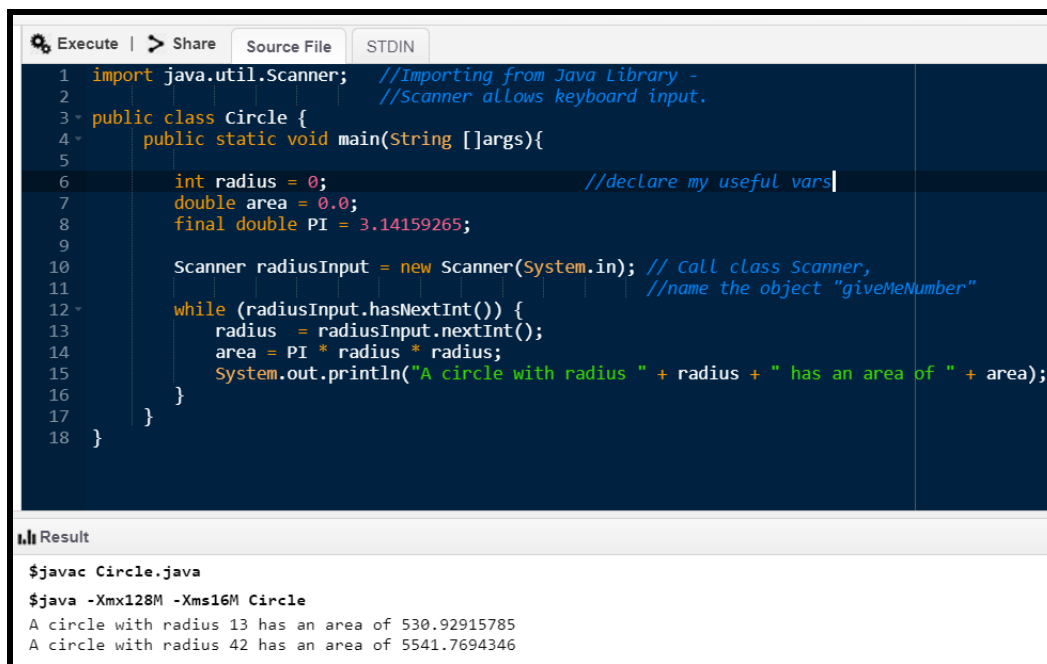
Result

```

$javac HelloWorld.java
$java -Xmx128M -Xms16M HelloWorld
(1,-1), (1,-1), (1,-1), (1,-1), (1,-1), (1,-1), (1,-1), (1,-1), (1,-1), (1,-1)

```

Figure 21. Hands-on practice. Challenge: repeat (1,-1) ten times; separate all phrases with a comma, but do not leave a trailing comma.



```

1 import java.util.Scanner; //Importing from Java Library -
2                             //Scanner allows keyboard input.
3 public class Circle {
4     public static void main(String []args){
5
6         int radius = 0; //declare my useful vars
7         double area = 0.0;
8         final double PI = 3.14159265;
9
10        Scanner radiusInput = new Scanner(System.in); // Call class Scanner,
11                                                    //name the object "giveMeNumber"
12        while (radiusInput.hasNextInt()) {
13            radius = radiusInput.nextInt();
14            area = PI * radius * radius;
15            System.out.println("A circle with radius " + radius + " has an area of " + area);
16        }
17    }
18 }

```

Result

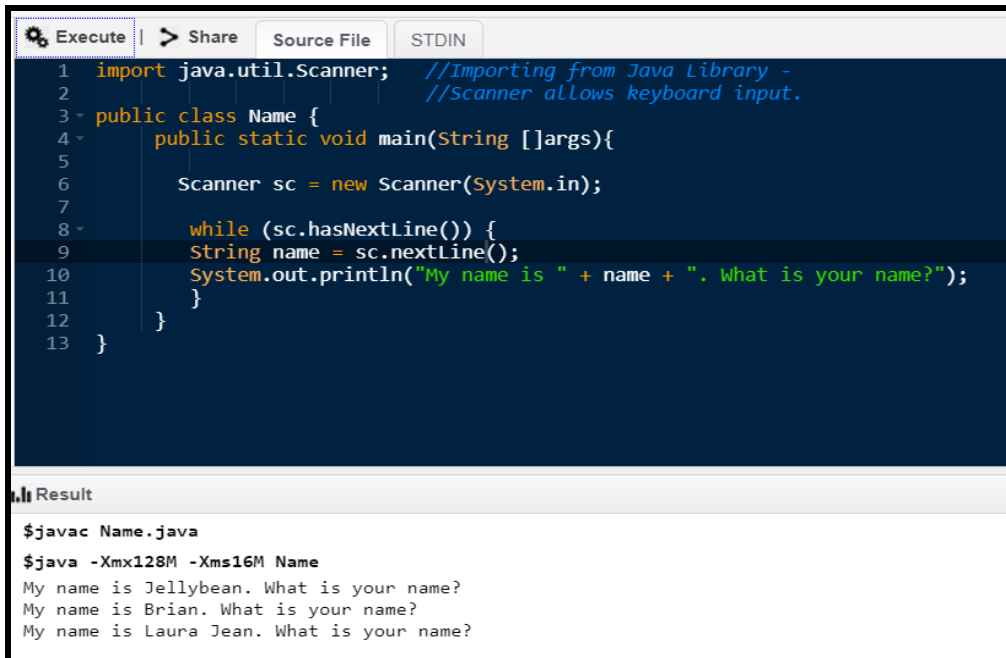
```

$javac Circle.java
$java -Xmx128M -Xms16M Circle
A circle with radius 13 has an area of 530.92915785
A circle with radius 42 has an area of 5541.7694346

```

Figure 22. Hands-on practice. Challenge: Read radius from the console input and compute the area and write to the screen. “Scanner” class refers to the protocols written at

<https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html>

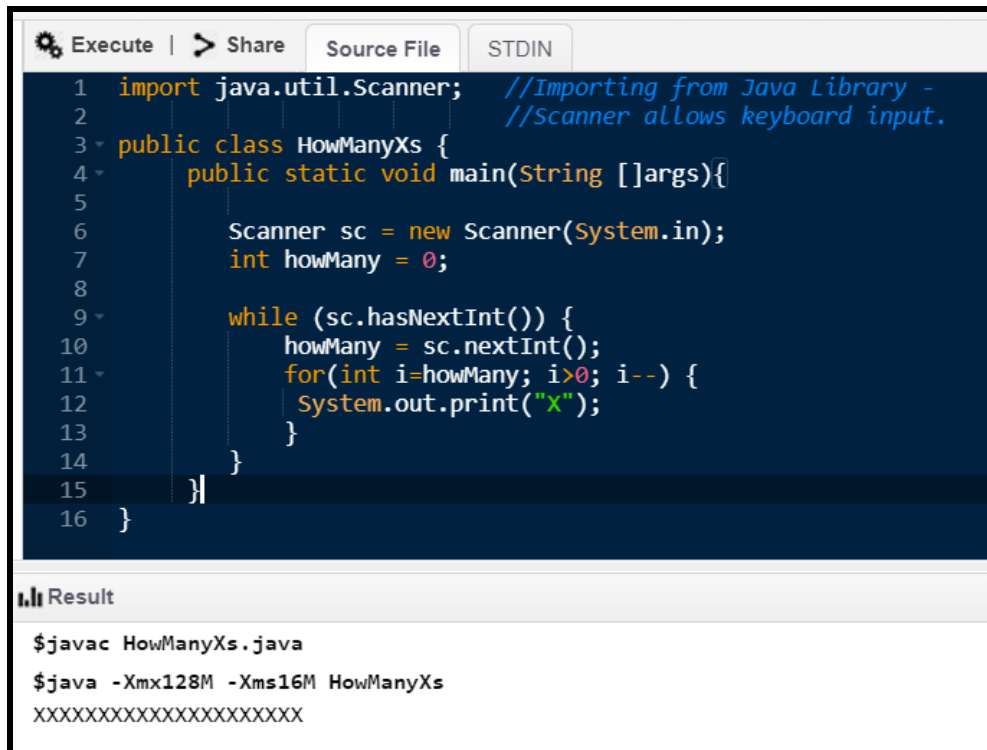


```
Execute | > Share | Source File | STDIN
1 import java.util.Scanner; //Importing from Java Library -
2 //Scanner allows keyboard input.
3 public class Name {
4     public static void main(String []args){
5
6         Scanner sc = new Scanner(System.in);
7
8         while (sc.hasNextLine()) {
9             String name = sc.nextLine();
10            System.out.println("My name is " + name + ". What is your name?");
11        }
12    }
13 }
```

Result

```
$javac Name.java
$java -Xmx128M -Xms16M Name
My name is Jellybean. What is your name?
My name is Brian. What is your name?
My name is Laura Jean. What is your name?
```

Figure 23. Hands-on practice. Challenge: Read a name (string) from console input and print it.



```
Execute | > Share | Source File | STDIN
1 import java.util.Scanner; //Importing from Java Library -
2 //Scanner allows keyboard input.
3 public class HowManyXs {
4     public static void main(String []args){
5
6         Scanner sc = new Scanner(System.in);
7         int howMany = 0;
8
9         while (sc.hasNextInt()) {
10            howMany = sc.nextInt();
11            for(int i=howMany; i>0; i--) {
12                System.out.print("X");
13            }
14        }
15    }
16 }
```

Result

```
$javac HowManyXs.java
$java -Xmx128M -Xms16M HowManyXs
XXXXXXXXXXXXXXXXXXXXXXX
```

Figure 24. Hands-on practice. Challenge: Type in an integer and print out that number of “X”s.

This module also covered information about networks. A network is simply a pathway--of objects, information, digital or physical--from one place to another. Generally, networks involve “protocol stacks”, which are essentially what layers of needs are involved in making a network functional. The simplified 5-layer model goes from Physical, to Data, to Network, to Transport, and finally to Application. Each layer has its own requirements, and therefore its own protocols. Protocols are “sets of rules”, including the methods and languages used within that layer.

### OSI Layers

- Physical Layer: The path over which data passes. Wired systems require twisted pair copper wire (Cat5e or Cat6) or microwave (wireless), Fiber, coaxial, cable, DSL
- Data Layer: Takes the “1”s and “0”s and somehow translates them in ways that they can pass over the physical layer and between the network and a console. *Ethernet* is by far the most popular protocol for the data layer. Hubs vs. Switches - Hub is a shared media device; everyone sees everyone else’s packets. Cheap, but less secure. Switches generally acts more as switchboards, routing messages directly to a destination.
- Network Layer: At this layer, devices are actually identified and addressed, and networks can organized subnetworks. Linked networks become “Internetworks”. There are several Network layer protocols no longer in use; the one still in use is “Internet Protocol”, or IP.
- Transport Layer: Isolates messages between lower & upper layers, quality control for messages. TCP and UDP protocols are at this layer. TCP is a host-to-host protocol,

establishes a connection, involves error checking and return calls, and provides some amount of confirmation & security. UDP, does not maintain a connection and so does not guarantee delivery (non-reliable).

- Application Layer: *(includes the sub-layers of Session, Presentation, & Application)* -
  - Session - establishes logical connections between systems for a certain time, manages log-ons, passwords, etc, and terminates the connection at the end.
  - Presentation - Format & Code conversion such as ASCII to BEDIC, encryption/decryption, compression. Prepares data from lower layers for presentation.
  - Application - Access to network for end user through mail, smtp, web, file transfer (specifically applications that need network connections)
    - TCP-IP applications use TCP protocol in the Transport Layer. TCP/IP programs include:
      - SMTP (Simple Mail Transfer Protocol)–Basic e-mail facility, transferring messages among hosts
      - FTP (File Transfer Protocol)–Sends files from one system to another on user command
      - Telnet–Remote login capability, allowing a user to emulate a terminal on the remote system



When browsing the web, we are using all of the OSI layers. The actual console you use (laptop, computer, playstation, etc), the wires or wireless waves the data is passing over, and the local router, all touch upon the lowest three layers (physical, data and network). TCP is what allows your local requests to make calls to a foreign network, somewhere on the internet. When actually using a browser such as Google Chrome, all the “Application” layer sublayers come into play. Every time you log in to a website you access the Security layer, including when using social media or email; this also comes into play while browsing certain websites, if they create a session in order to cache information such as a temporary shopping cart. The presentation layer translates data between different coding and protocol languages, and handles server-side processes. The application layer is what actually transfers data to the user, and that is what we interact with visibly.

A separate set of protocols are used at every OSI layer. The physical layer uses Cat5 cables, microwave waves for wireless, coaxial, or other cables. The data layer uses ethernet cable and ethernet packets. The network layer uses IP and MAC addresses for identity. The transport layer uses either TCP (more secure connections) or UDB (non-reliable connections). Applications that are TCP/IP and use the TCP protocols include HTTP (web), SMTP (email), FTP (file transfer), and Telnet (remote login).

The module wrapped up with all participants receiving a Raspberry Pi 3 and learning network commands that can be executed through a command line on a computer or through Raspberry Pi. We also were able to construct and plug a few Raspberry Pis into monitors to play with the pre-installed software. Our Pis came with Scratch and BlueJ Java IDE pre-loaded,

among other programs. Figures shown below were taken during practice sessions with the command line.

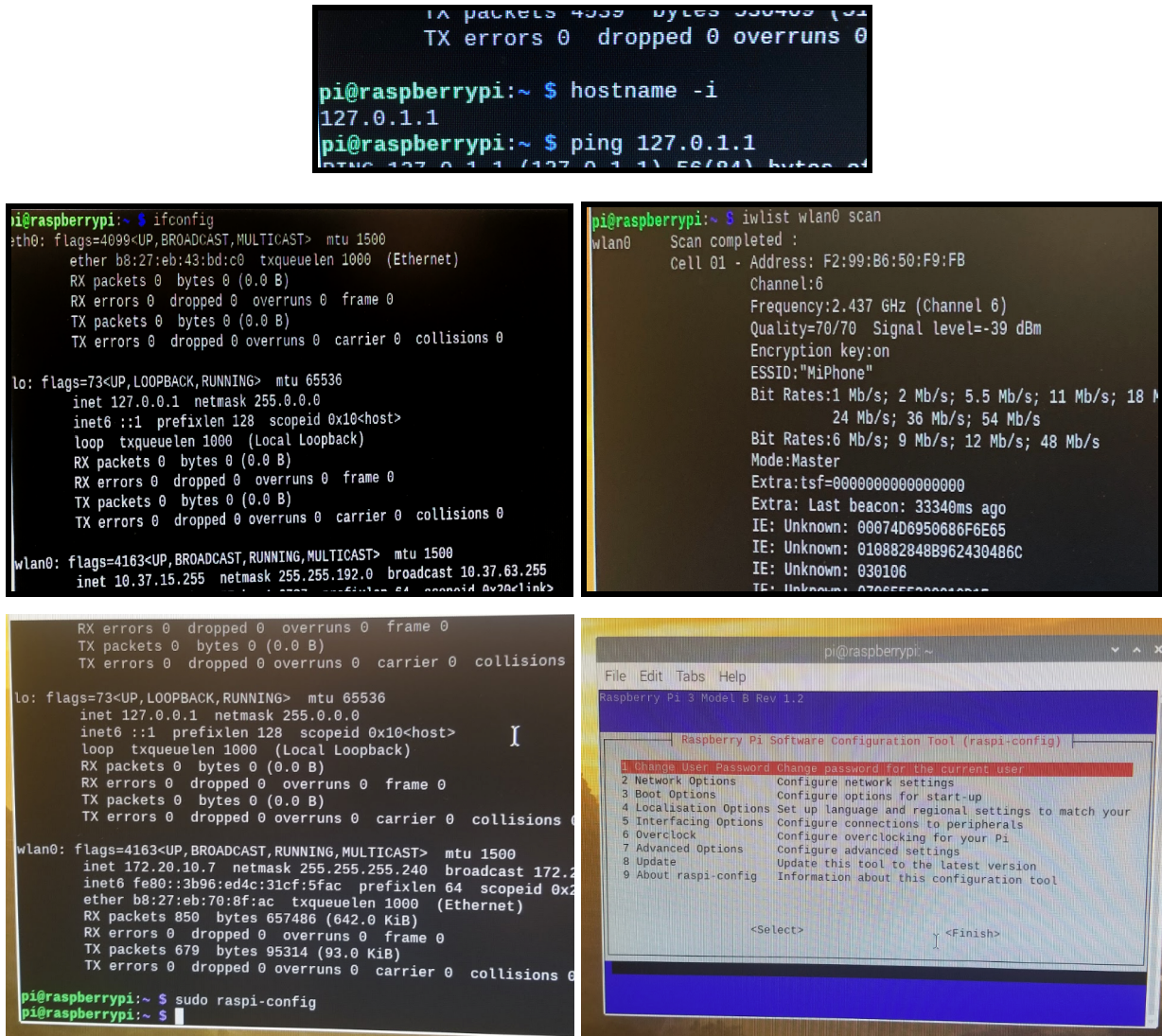


Figure 25. Command line practice in Raspberry Pi. *Top:* Simple hostname local IP lookup. *Middle:* advanced IP lookup and WLAN Wireless status. *Bottom:* Config command and config screen.

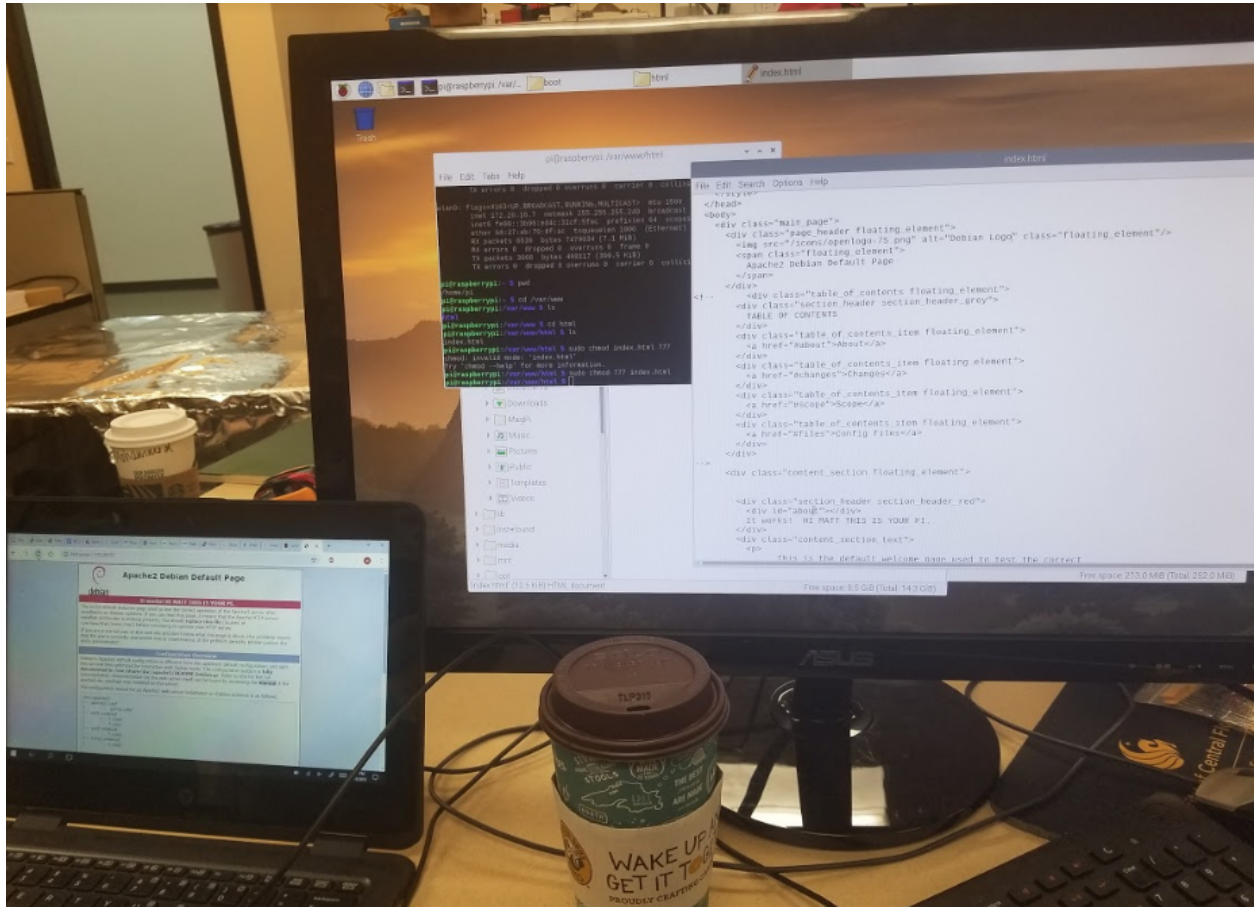


Figure 26. Apache Server enabled, with introductory web page displaying on a separate computer.

## Translation to Classroom Teaching

There are several connections between RET experience and chemistry classroom teaching, mainly in two categories: material science (properties of substances such as CNTs and relationship with molecular arrangement) and scientific modeling (the use of programming tools to create visual and conceptual models and calculators).

## Carbon Nanotubes and Material Science

The connections between Carbon Nanotechnology and chemistry are numerous. Topics that we touched upon included properties of carbon atoms (SC.912.P.8.5), molecular shape (graphene arrangement), chirality (directionality of the carbon rings within the nanotube), physical and chemical properties of the material based on chirality, intermolecular forces (covalent bonds and aromatic graphene rings), intramolecular forces (Van der Waals forces, SC.912.P.8.6), material properties of substances and alloys/composites, transformation of energy during labs (SC.912.P.10.1), and more. I could certainly use portions of my experience to describe additional examples during a lecture.

- Properties of Carbon - Special properties of Carbon vs Silicone as semiconductors - <https://electronics.howstuffworks.com/diode.htm>
- Periodic Table Trends - Metals vs Transition metals vs nonmetals in manufacturing
- Gas laws - Strain sensors as measurement tools for Gas law applications ( $PV=nRT$ ,  $PV$  is often atm,  $1 \text{ atm}=101,325 \text{ Pascals}$ ,  $1 \text{ Pascal} = 1 \text{ Newton/sq meter}$ , Strain sensors can

detect strain which can be converted to Newtons - and detect changes in volume in an irregular shape like an oval balloon)

### **Molecular arrangement, chirality and characteristics.**

The way a molecule or matrix is arranged will affect its characteristics. To demonstrate this, a kinesthetic activity might be one in which a group of students - say, groups of five - are told they must “bond” into a matrix with one another (every person should be bonded - in physical contact with - two other people, and all must be bonded in the same way). They could be tasked with being able to perform a variety of tasks, for which some “bond configurations” would be more effective than others.

- Some properties might be:
  - passing an electron [ball] around to each atom [person] - electrical conductivity
  - all atoms being able to dance - thermal conductivity
  - resistance to being deformed by the application of gentle pressure - structural stability / strain resistance
  - ability to change shape - malleability
  - ability to stretch into a long shape - ductility
- Possible limitations to this activity might include some students’ reluctance to participate in a contact activity, and the possibility of injury if students are too aggressive in the demonstration for structural stability. Because of this, it might be a good idea to choose

only a handful of students to demonstrate for the whole class, and for stress/strain to be applied by teacher in only a superficial way, perhaps using a beach ball.

- An extension of this activity could be done with hexagonal graph paper to demonstrate isomerism (i.e. try to draw the same molecular formula in 10 different ways; or, using scissors and tape, try to determine the three “Chiralities” of a carbon nanotube, and infer the properties of each.
- A non-kinesthetic version of the above could also be done by providing students with a material - such as yarn - to perform the activity. Students could be given a minimum and maximum amount of yarn for the “carbon nanotubes”, as well as “substrates” (sticks/rulers/pool noodle foam), “matrix” (glue), “composites” (other materials like marbles, pipe cleaners, etc), as well as tools such as scissors. Students could then attempt to compete in several different categories, such as:
  - electric conductivity (ability to pass a ping pong ball 2 feet)
  - thermal conductivity (how much movement or vibration on one piece of the material moves the remainder of the material)
  - strength (inability to be broken under certain circumstances - such as laying a heavy ball on it, or applying pressure from a beach ball, etc)
  - flexibility (ability to be bent or folded)
- *Note: Inspiration for this activity came in part from a 6th grade lesson plan in which students model the movements of water molecules in different phases (Nance, n.d.)*

**Unit analysis / Solving for unknowns** - Analyzing formulas such as the Gauge Factor

$$\frac{\Delta R}{R_0} = k \cdot \mathcal{E}$$

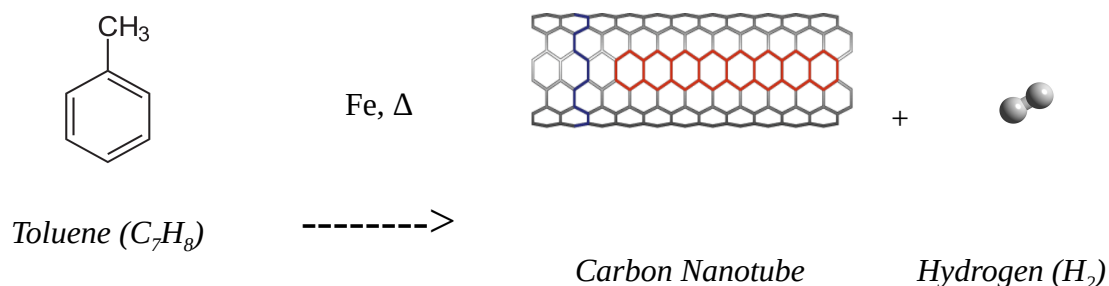
$\Delta R$  = The SG's change in resistance due to the strain  
 $R_0$  = The SG's nominal resistance  
 $k$  = The SG's gauge factor (SG sensitivity)  
 $\mathcal{E}$  = The strain to be measured

1. Isolate variables - asking students to “arrange to solve for k”
2. Asking students to determine “which is the most sensitive sensor?”
  - a. Concept being - if students can get used to solving symbolic formulas they will have less trouble in later stoichiometry / gas laws / thermodynamics units, as well as in Physics & higher Mathematics

**Intermolecular forces** - Ask conceptual questions about Carbon intermolecular bonding

3. *Question:* “Why don’t carbon nanotubes dissolve in water?”
  - a. *Answer:* Van der waals forces! Weak individually but strong all together
4. *Additional question:* What could we do to make it more soluble in water?
  - a. *Answer:* Make it polar! (Add functional groups)

**Stoichiometry** - Develop a few questions based on creation of carbon nanotubes.



1. “In the reaction with Toluene, ferrocene is used as a catalyst - a source of iron. Because of the method of synthesis, a solution of the two substances must be mixed beforehand, 97.5% Toluene and 2.5% Ferrocene (by weight).
2. We measure 50mL of liquid Toluene. Toluene has a density of .867 g/mL. How many grams of Toluene do we have?
3. If Toluene should be 97.5% (by weight) of the mixture, what should the total mass of the mixture be?
4. How much ferrocene should we add to the solution to ensure the best yield (remember 2.5% by weight should be the catalyst)?

### Small Program for Writing Chemical Formula of Metal Salts

It would be possible for me to introduce some basic coding concepts using a software such as code.org, and ask the students to accomplish some tasks. For example, one of the common tasks required of high school chemistry students is to predict chemical formulas of salts (molecules with ionic bonds) based on the charges of the participating ions. Many computer-based activities could be created based on this standard:



- Create a table of common ions and their charges.
- Record a few examples of an anion and cation combining, and the resulting chemical formula.
- Write a logical statement (instructions, in words), on how to construct a chemical formula. Use ONLY the terms: \*\*\*\*\* *these could be altered to more user-friendly, conversational words*
  - variables: “cation”, “anion”, “charge”, “subscript”
  - logic: “if”, “then”, “else”, “and”, “or”, “not”
  - commands: “print” (or “add to the formula...”)
- Code a simple program to create the chemical formula by the person entering in each of the variables themselves - *Can use Scratch block-based OR alter pre-written code in Java, Python, or Excel*
- Challenge students to plan additional features to their program - what could improve it? *If they are capable, offer extra credit for them to attempt to actually write the program.*

Some ideas they might have could be:

- The student creates a table or array with the possible ions and their charges; the program references it so that the user only needs to select two ions.
- The program simplifies chemical formulas when the subscripts have a common factor.

- The student creates a table with the possible ions and their groups from the periodic table; the program references it against a table of group charges. Again, the user only needs to select two ions, but two tables and/or arrays are referenced.
- The program accounts for transition metals with more than one charge state, either by printing all possible chemical formulas, or by requesting more information from the user.
- The program gives a warning when invalid input is attempted (two anions or two cations).
- The program prints information about the molecule (properties, pictures, links to wikipedia, etc) along with the chemical formula.

### **Stoichiometry Calculator (Excel or Scratch)**

Building on other coding projects and scaffolded processes, it would be possible to guide students to create calculators to perform common math operations that they find challenging. In theory, students writing their own calculation functions will help them to visualize the simplicity of the operation, and allow them to see the operation procedurally.

- Programmable concepts in this unit could be things like:
  - Convert between moles and liters of gas (very easy, multiply by whole number constant)
  - Convert between moles and particles (easy, multiply by constant with scientific notation)

- Convert between moles and mass (medium to difficult, multiply by another variable - either input by user or lookup from alternate sources)
- Convert between moles of reactant to moles of reactant in a specific reaction (easy to difficult depending on if the teacher has pre-programmed the associated chemical reaction)
- **Possible mediums** for rudimentary programming and/or modeling logical processes would include: Scratch, Code.org, Excel, or even rearrangeable physical foam blocks with statements written on them. *I have already tried and succeeded modeling a solution calculator for the ideal gas law,  $PV=nRT$ , on each of these methods - they are all possible with minimal learning curve.*
- *Note: I will be investigating further possibilities based on some ideas I've harvested from online research, such as a post on "Chem Ed Xchange" that suggests creating visual models on [www.glowscript.org](http://www.glowscript.org) using [VPython](#), or using Python (basics can be learned through [codecademy](http://codecademy)) and [Cloud9](#) to create calculators (Stewart, 2016). Many resources I studied also made liberal use of the [Jupyter Notebook](#).*

**Connections with Scratch Block-Based Visual Programming Language** - The connection between coding and student motivation is obvious; providing students with a digital playground with which to build unique visuals and interactive models not only ties in to "Nature of Science" as well as engineering content, but it is a fun and engaging activity that is freely accessible.

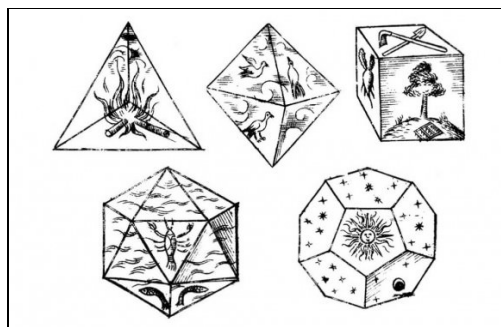
Normally, it would not be possible in the scope of a normal curriculum to incorporate enough lesson time to teach coding; however, the program "Scratch" provides such a low floor to

entry that students could begin designing their own models in one or two class periods. Model creation is an important part of introductory chemistry; Chemistry as a science is so difficult to convey because most of it occurs on a nanoscale and is therefore invisible to students. One of the first things, then, that students should learn is how models are used--and making their own is a great way of doing just that.

One of the first projects I'd like to assign in scratch is to have students animate one of the early atomic models. They could choose:

1. Dalton's OR Democritus's Indestructible Particles

- a. *Everything is made of atoms, which are indestructible tiny particles that are constantly in motion. There are countless types of atoms, a different type for every material, and they all differ in shape and size (Williams, 2015).*

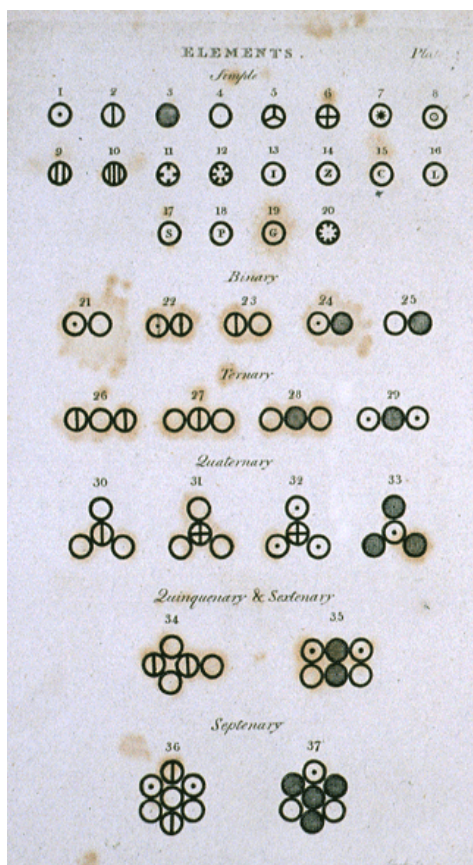


*Above Image (Williams, 2015);*

*Below Image (Ball, 2016)*

- b. *Some believed atoms of one element could be changed into atoms of another (i.e., alchemy), while others did not. Dalton believe all atoms were some form of sphere & used wooden balls to represent them (Ball, 2016)*

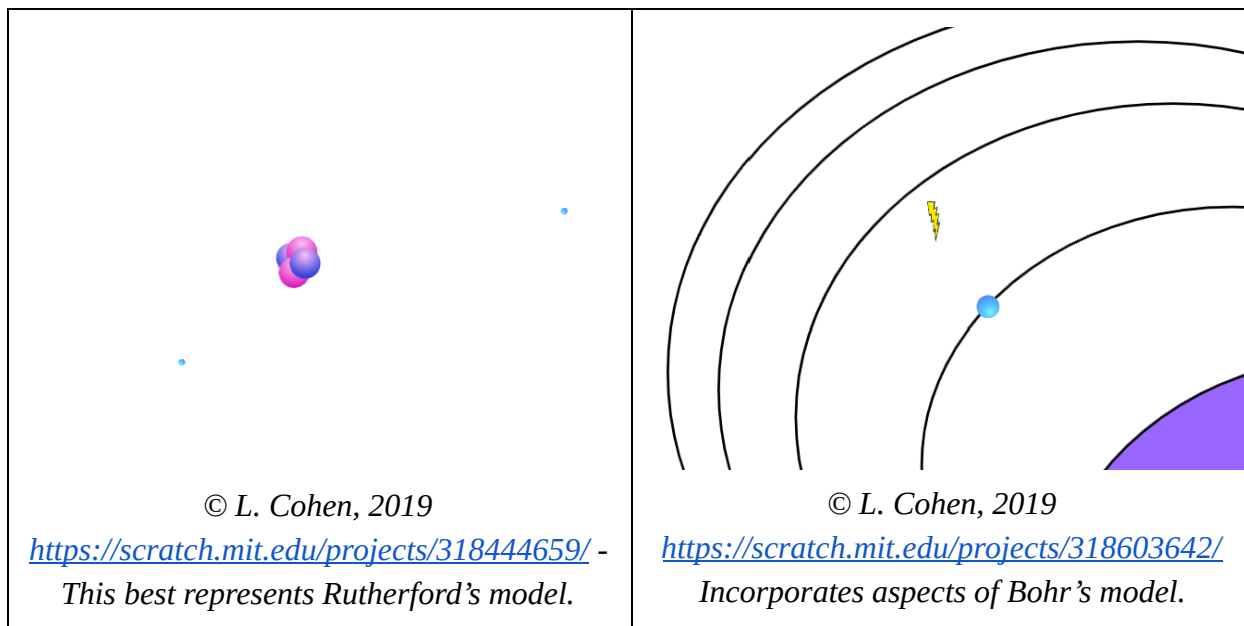




(Dalton, 1808)

2. Plum pudding Model - negatively charged particles floating in a positively charged matrix, imagined as raisins in pudding, or chocolate pieces in a cookie.
3. Rutherford Model - A tiny positive nucleus with electrons randomly moving around it.
4. Bohr Model - Electrons move like planets in orbitals a fixed distance from the center - they can jump from one orbital to another, but can't be between orbitals.
5. Quantum Atomic Model - Electron locations are probabilistic, and electrons act much like light--partly particles, partly wave-like. This model also adds the distinct shapes of different suborbitals.

I created two examples of an atomic model I could show students as a jumping-off point:



## Lesson Learned

From my experience in the RET: Comet Program, I gained an understanding of IoT and an appreciation for the variety of disciplines which are involved in developing the devices, connections and protocols which comprise the IoT networks. I've also learned that the type of world my students will be entering as adults will be vastly different than the one I entered into from high school and college; it will be a world that expects them, at the outset, to be *more than* digital natives. My students' world will expect them to be digital *creators*, and to use their lifelong digital literacy to contribute to their communities and to their employers in meaningful ways.

To this end, it is my responsibility as an educator to go beyond the superficial goal of “integrating technology” into the classroom, and to instead promote deep, meaningful digital literacy. This means doing more than assigning homework through an online class management system, or directing students to do interactive multiple choice questions on popular learning websites. The type of digital literacy that my students will most benefit from is that which encompasses the concepts of programming. Regardless of the language used (be it Java, Python, or a visual block-based language such as Scratch), these are all derived from the language of logic, and the learning of one opens the doors to all others. Exposure to programming allows people to participate in an increasingly computerized culture, a culture which will change our lives in ways we can only imagine now.

### Works Cited

- Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17, 2347–2376. doi:10.1109/comst.2015.2444095
- Ball, P. (2016). In retrospect: A new system of chemical philosophy. *Nature*, 537(7618), 32.
- Carr, K. (2014, July 17). Computer science adds new dimension to study of chemistry. *The Stanford Daily*. Retrieved from <https://www.stanforddaily.com/2014/07/17/computer-science-adds-new-dimension-to-study-of-chemistry/>
- Dalton, J. (1808). *A new system of chemical philosophy*, Volumes 1-2. Manchester: R. Bickerstaff, Strand, London.
- Digilent. (n.d.). Basys 3 Artix-7 FPGA Trainer Board: Recommended for Introductory Users. Retrieved from <https://store.digilentinc.com/basys-3-artix-7-fpga-trainer-board-recommended-for-introductory-users>
- Ebbert, J. (n.d.). CodingBat: Java. Retrieved from <https://codingbat.com/java>
- Gou, J., Tang, Y., Liang, F., Zhao, Z., Firsich, D., & Fielding, J. (2010). Carbon nanofiber paper for lightning strike protection of composite materials. *Composites Part B Engineering*, 41(2), 192-198. doi:10.1016/j.compositesb.2009.06.009
- Gullen, K., & Sheldon, T. (2014). Synergy sparks digital literacy: Redefined roles create new possibilities for teachers and students. *The Learning Professional*, 35(2). Retrieved from



<https://learningforward.org/journal/april-2014-issue/synergy-sparks-digital-literacy/>

- Hu, C., Li, Z., Wang, Y., Gao, J., Dai, K., Zheng, G., . . . Guo, Z. (2017). Comparative assessment of the strain-sensing behaviors of polylactic acid nanocomposites: reduced graphene oxide or carbon nanotubes. *Journal of Materials Chemistry*, 5(9), 2318–2328. doi:10.1039/c6tc05261d
- Jeon, B., Jeong, B., Jee, S., Huang, Y., Kim, Y., Ho Park, G., . . . Hyun Choi, T. (2019). A facial recognition mobile app for patient safety and biometric identification: Design, development, and validation *JMIR Mhealth Uhealth*, 7(4), e11472. doi:10.2196/11472
- Li, C., Zhang, D., Deng, C., Wang, P., Hu, Y., Bin, Y., . . . Pan, L. (2018). High performance strain sensor based on buckypaper for full-range detection of human motions. *Nanoscale*, 10(31), 14966-14975. doi:10.1039/c8nr02196a
- Maloney, J., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. Paper presented at the SIGCSE'08 - Proceedings of the 39th ACM Technical Symposium on Computer Science Education.
- Michel, G. (2015). US Patent No. US9072555B2.
- Moreno-León, J., Robles, G., & Román-González, M. (2016). Code to learn: Where does it belong in the K-12 curriculum? *Journal of Information Technology Education: Research*, 15, 283-303. doi:10.28945/3521
- Nance, P. (n.d.). Movement of water molecules at different temperatures during the three states of matter. Retrieved from <http://elemscience.jordandistrict.org/files/6.2.2.1-Kinesthetic-Movement-of-Molecules.pdf>

- National Instruments. (2019, May 24). Measuring strain with strain gages. Retrieved from <http://www.ni.com/en-us/innovations/white-papers/07/measuring-strain-with-strain-gages.html>
- National Research Council. (2012). A Framework for K-12 Science Education: Practices, Crosscutting Concepts, and Core Ideas(pp. 400). doi:doi:10.17226/13165
- Prensky, M. (2001). Digital natives, digital immigrants. *On the Horizon*, 9(5). Retrieved from <https://www.marcprensky.com/writing/Prensky%20-%20Digital%20Natives,%20Digital%20Immigrants%20-%20Part1.pdf>
- Pyrograf. (n.d.). Pyrograf-III carbon nanofiber, stacked-cup carbon nanotubes. Retrieved from [http://pyrografproducts.com/nanofiber.html#\\_PR-24-XT-HHT\\_Data\\_Sheet](http://pyrografproducts.com/nanofiber.html#_PR-24-XT-HHT_Data_Sheet)
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60-67. doi:10.1145/1592761.1592779
- Stewart, L. (2016, March 23). Coding in chemistry. *Chemical Education Xchange*. Retrieved from <https://www.chemedx.org/blog/coding-chemistry>
- Velez Rueda, A. J., Benítez, G. I., Marchetti, J., Hasenahuer, M. A., Fornasari, M. S., Palopoli, N., & Parisi, G. (2019). Bioinformatics calls the school: Use of smartphones to introduce Python for bioinformatics in high schools. *PLoS Computational Biology*, 15(2), e1006473. doi:10.1371/journal.pcbi.1006473
- Wang, X., Sparkman, J., & Gou, J. (2017). Strain sensing of printed carbon nanotube sensors on polyurethane substrate with spray deposition modeling. *Composites Communications*, 3,

1-6. doi:10.1016/j.coco.2016.10.003

Wang, Z., Liang, Z., Wang, B., Zhang, C., & Kramer, L. (2004). Processing and property investigation of single-walled carbon nanotube (SWNT) buckypaper/epoxy resin matrix nanocomposites. *Composites, Part A, Applied Science and Manufacturing*, 35(10), 1225–1232.

Williams, M. (2015, December 11). Who was Democritus? *Universe Today*. Retrieved from <https://www.universetoday.com/60058/democritus-atom/>

Woo, D. T., Hudson, B. T., Mori, J. C., Ngan, E. S. M., Pak, W.-Y., & Haines, R. S. (2007). Interdisciplinary educational collaborations: chemistry and computer science. *Journal of Chemical Education*, 84, 967–970. doi:10.1021/ed084p967

Zhu, R., Yamamoto, I., Lawn, M. J., Hashimoto, Y., Nagayasu, T., Yamasaki, N., & Matsumoto, K. (2017). Research and development of a laparoscopic surgical device for ligating endless organs based on a flexible structure. *Computer Assisted Surgery*, 22, 36-44.  
doi:10.1080/24699322.2017.1378790