October 2015

Physical Games for Learning

Alex Wayne Silva
Worcester Polytechnic Institute

Christopher James Cerruti
Worcester Polytechnic Institute

Dalton Scott Tapply
Worcester Polytechnic Institute

Dylan Wadsworth McCarthy
Worcester Polytechnic Institute

Peter Arthur Leondires
Worcester Polytechnic Institute

See next page for additional authors

Physical Games for Learning

An Interactive Qualifying Project

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

May 4th, 2015

Report Submitted to Professor Ivon Arroyo

Chris Cerruti

Peter Leondires

Dylan McCarthy

Shadi Ramadan

Alex Silva

Dalton Tapply

Contents

1. Abstract

The goal of this project is to teach elementary mathematics and number sense through the use of a cyber-watch or smartphone. The device enables an alternate learning style fueled by the use of interactive games and other technologies. The hope is to augment the learning process for students that do not learn well in a traditional classroom by providing a fun and engaging learning style. This research improves on the original idea presented by Professor Ivon Arroyo, modifying the alternate learning experience to help students become more motivated about learning mathematics, as well as providing real world applications to in-class materials without the monotony of typical classroom teaching. Blending wireless wearable technology, educational games, mathematics, and real time reporting could very well lead to accelerated learning.

The outcome of this project is a novel infrastructure, which allows for the creation of a myriad of interactive embodied learning experiences for students of all ages.

2. Background and Literature Review

2.1 NSF Cyberlearning: Cyberhoodies, by Ivon Arroyo

We started our readings with a grant proposal submitted to the Cyberlearning program at the National Science Foundation that summarizes the work done in previous years at WPI on E-textiles, or Electronic Textiles for Education. One of the major advantages of E-textiles is that they blur the lines between home and school, something desirable for the future of Educational Technologies as specified in a “Roadmap of Educational Technologies” (Woolf, 2010). By providing students with a learning opportunity that incorporates physical activity, learners who typically have decreased attention span for seated classroom activities may find themselves more interested (Arroyo, p. 3). This combination of physical activity with real-world learning experience significantly augments the learning experience and increases number sense without making students feel as though they are participating in a stereotypical learning activity (Arroyo, p. 5).

A few of the learning games facilitated by the E-textiles were hide and seek, scavenger hunt, and cyber game (a technological spin-off of hide and seek). These activities engaged students’ real world number sense by instructing them to locate objects a certain distance from other landmarks or of a certain size. After participating in both hide and seek and scavenger hunt pilot games, students ranked various aspects of the games on a scale from 1-5 (5 being the highest). The study summarized in the document showed that while students enjoyed both games they were largely unaware of the academic value of the activities. They responded to the question “do you think you learned math playing this game” with an average of 2.67 out of 5 between each group (Arroyo, p. 10). This presents a problem, as one of the major goals of this project is to increase the level of student interest in mathematics. If students fail to see the correlation between classroom learning and the physical activities facilitated by the E-textiles, they won't associate their interest in the activities with general math skills, leading to a missed opportunity to spark interest in mathematics in today's young learners.

2.2 Embodiment in Mathematics Teaching and Learning: Evidence From Learners' and Teachers' Gestures, by Alibali and Nathan

Another way that we can qualify that students are learning is by facilitating and observing their use of gestures. Gestures are often taken as evidence that the body is involved in thinking and speaking about the ideas expressed in those gestures (Alibali & Nathan, 2012, p. 247). Alibali and Nathan suggest that there are four modes of gesturing:

(a) *pointing gestures, which are gestures that serve to indicate objects or locations*

(b) iconic gestures, which are gestures that depict semantic content directly via the shape or motion trajectory of the hand

(c) metaphoric gestures, which depict semantic content via metaphor (e.g,cupping hands as if to "hold" an idea)

(d) beat gestures, which are rhythmic gestures that do not express semantic content but that instead align with the prosody of speech

(Alibali & Nathan, p. 247)

Of these four modes, they focus mainly on pointing, iconic, and metaphoric gestures.

Pointing gestures are used as a method to physically link speech and associated mental processes to the physical environment (Alibali & Nathan, p. 253). In other words, when a student uses pointing gestures they are assigning a physical or visual meaning to what they are being taught. An example figure for pointing gestures was provided by Alibali and Nathan below.

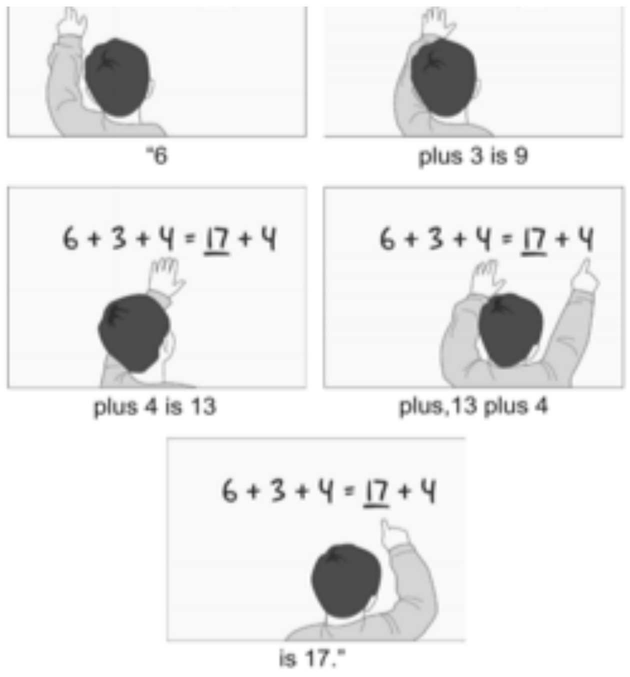


Figure 1: Pointing in a student's explanation of a mathematical equation.
Extracted from (Alibali & Nathan, p. 256)

Iconic gestures occur because thinking is based in perception and action. Thus, representational gestures provide support for the claim that cognition is based in the body (Alibali & Nathan, p. 254). In other words iconic gestures help us notice that students are learning, as well as help teachers teach, because of the assigning of an action to a certain situation. This helps students recall their teachings by assigning a physical action to a problem. Alibali and Nathan also provide a graphic for this type of gesturing, shown in Figure 2.

Figure 2: Representational Gestures. Extracted from (Alibali & Nathan, p. 261)

Metaphoric gestures are used because aspects of human experience are inherently spatial and therefore readily expressed in gestures. For example, Health is up and sickness is down (Alibali & Nathan, p. 255). These gestures are very similar to iconic gestures; the difference is that the gesture itself is based in a metaphor, not on a representational action. Again, Alibali and Nathan provide a graphic to help understand this type of gesturing, illustrated in Figure 3 below (p. 269).

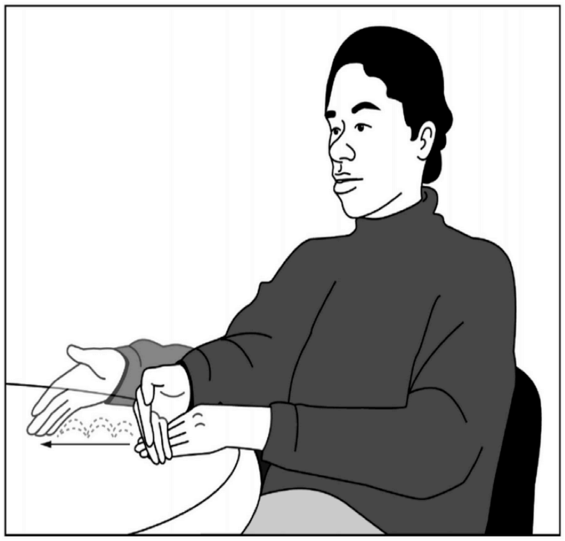


Figure 3: Metaphoric Gesture. Extracted from (Alibali and Natan, 2012)

By observing the students during the case studies such as the tangrams race and Estimate It!, we can tell whether or not the students are using gestures to help solve the problems presented by the cyber watch. The science of gesture based learning is hard to quantify since it is based purely off of observation and is hard to test. By studying and keeping a video record of the case studies we can hopefully provide more insight to the science of gesture based learning and teaching.

2.3 ZUL: A Light-weight Architecture for Zigbee-based Ubiquitous Applications, by Imran Zualkerman

One of the ideas that was presented and used as inspiration for an early iteration of the cyber watch design was a Zigbee-Based Ubiquitous Layer, nicknamed ZUL (Zualkernan, 2011). This system would be used as a way for testers to simply apply a catalog of questions to an end node like a screen. This happens through the use of sensors that input to the host and then to the internet where it can be stored and analyzed at a later date. There are some restrictions in regard to this system, a few being:

a) Host and end node processing require a large amount of resources

b) The system is asynchronous when receiving input information and exporting that information to the host and the only way for a signal to be sent is through event-driven timing

The software associated with ZUL technology consists of two virtual machines designated to be the host system and the Zigbee-node transceivers. Both of the virtual machines are designed in the C programming language and .NET framework. One of the systems is a ZUL-H system designed for the host, allowing the host to send and receive information based on the results of the nodes. The nodes use a ZUL-E system for easy translation to the host and to interpret the signals from the ZUL-H system. Finally, the entire system together consists of: a group of co-processor configured Zigbee nodes, two 8-bit RISC architecture low-power microcontrollers in master-slave mode, and the output systems such as buzzers and lights.

2.4 Constraining Movement Alters the Recruitment of Motor Processes in Mental Rotation David Moreau

Some of the readings we did corresponded to understanding the importance of embodiment for performance and cognition, trying to answer questions such as: “Does mental rotation depend on readiness to act?” Moreau suggests that different levels of motor ability may result in different strategies to solve problems or that motor processes in mental rotation are experience-dependent (Moreau, 2012, p. 447). In order to test this claim Moreau ran a test with a control group of non-athletes and a group of nationally ranked Wrestlers (more mechanically inclined). He claims that movement restriction should result in detrimental performance for all individuals when mental rotation involves body items and that, when presented with non-body items, only motor experts should be affected, as they rely on motor recoding of abstract shapes. In short, how important is the readiness to act when attempting to solve mental rotation problems?

The test consisted of the two aforementioned groups. The first group was a control that consisted of 7 females and 9 males who did not practice any sport or physical activity. The test group consisted of 7 females and 9 males who practiced wrestling at an elite level (p. 449). Participants were sat in front of a computer screen, either bound by ribbon ties or allowed to have their hands free. They were shown a real object, either a hand model or a three dimensional shape, that they would have to identify from a set of rotated images on the computer. Sample images were provided on page 449 and shown below.

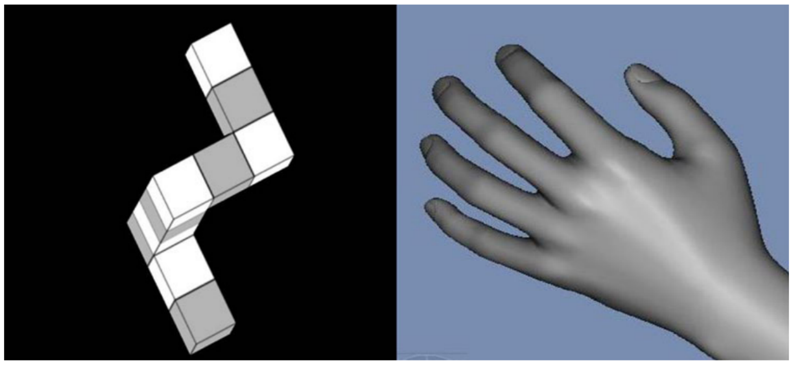


Figure 4: Computer Images (Moreau, p. 450)

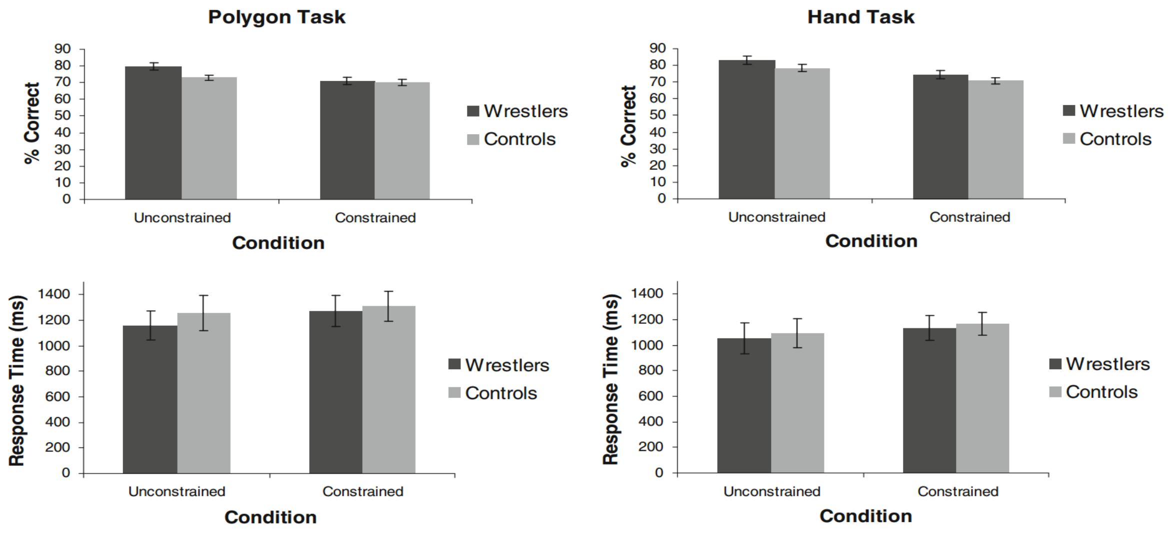


Figure 5: Results graphs comparing correctness and response times between the control group and the wrestlers. Extracted from (Moreau, p. 450)

From the results above we can conclude that mental rotation and visualization is indeed hindered by being constrained and not allowed the freedom to act, having motor activity unavailable to them. We may also conclude that the test group of wrestlers, who are more mechanically inclined, were both faster in response time and more correct in the mental rotation task. This paper suggests a clear association between cognition, performance and motor activity, as if ideas were encoded in motor form and we need motor action to encode and retrieve concepts related to geometry and visual rotations.

2.5 Immersive simulation for Smart Classrooms: Exploring Evolutionary Concepts in Secondary Science, by Lui and Slotta

We believe that having students learn through activities promotes learning and greater retention of knowledge. In their paper, Lui and Slotta discuss an application of this facilitation through the use of an ‘EvoRoom’ which consisted of networked tablet computers working with servers and projectors to produce an immersive environment (Lui and Slotta, 2014, p. 3). Students would register with the room so that the software agents would be able to track interactions with other students and the materials included in the room. Once all students had logged into the room, the instructor would assign the role of ‘field researchers’ to groups instructed to complete tasks. All real time updates would be shown on a board visible to the entire class (Lui and Slotta, p. 16). This process of completing tasks is most easily accomplished through cooperation between the instructor and the research team. Cooperation enables the research team to hand off the ‘ownership’ of the instructional design. Getting a complex system such as this one to work fluidly takes time and requires contribution from all parties. This activity took a cumulative six months to design the first trial and another six months to redesign the activity and perform it a second time (Lui and Slotta p. 19). The results of this trial were heavily in favor of the use of tablets and being in an immersive environment produced by the EvoRoom. For the use of tablets the average ratings were 8.5 and 7 out of a possible ten for iterations one and two respectively. The EvoRoom itself received an average score of nine for both iterations. All responses were taken as post activity questionnaires given out to the students participating on each day. These scores were students’ perceptions and do not relate to how well the material was understood by the students. Performance data was obtained through an independent posttest based on the material in the activity. The posttest yielded much higher results (average of 78.75 with a standard deviation of 16.16) compared to a pretest on the same material (average of 59.40 with a standard deviation of 19.00). This indicates an improvement in the students understanding of the material (Lui and Slotta, p. 24). The final findings of this paper were that the use of personal tablets was beneficial to learning but students still required some form of scaffolding to encourage student collaboration (Lui and Slotta p. 30).

3. First Prototype

For the first stage we needed to create a prototype that would be able to facilitate one of the games, such that we could determine the effectiveness and usability of our device. The result of this first prototype was a smart-watch style device that allowed for the game play of one of the simpler games, Tangrams Race. This stage did not require any communication between devices, and the games were simply pre-programmed onto the set of devices.

3.1 Goals

Our major goal was to redesign the old hoodie/backpack E-Textiles devices into something that was more appealing, safer, and robust. We hypothesize that the use of this more attractive and smaller intuitive design will enable younger students to use the device as well as spark their interest in the games being played, and technology in general. We also believe that our design will be more accommodating in that it will fit a larger number of students, without restrictions of age or body size.

3.2 Unit Design

The original designs were two-fold:

a) A backpack style device with all of the key components placed on a backpack with a display on the sleeve for interaction, based completely on textiles and the Lilypad Arduino microprocessor

b) A cyber-hoodie design, containing a display within the sleeve and wires running along the sleeve to the back where the majority of the hardware would be placed.

In this case the backpack is preferable as it is easier to accommodate students of varying age groups, given that a single backpack should fit the majority of children. The drawback in this case is that the components are very exposed, making them more susceptible to damage or potentially harmful to the children using them. We felt this design was outdated and could be improved and simplified down to the size of a large watch with all of the components consolidated underneath the display. The goal was to design a watch-like device that was small, easily adjustable, and better protected the components. To make this design change, Shadi, Alex, and Dylan worked on a SolidWorks model for the new device.

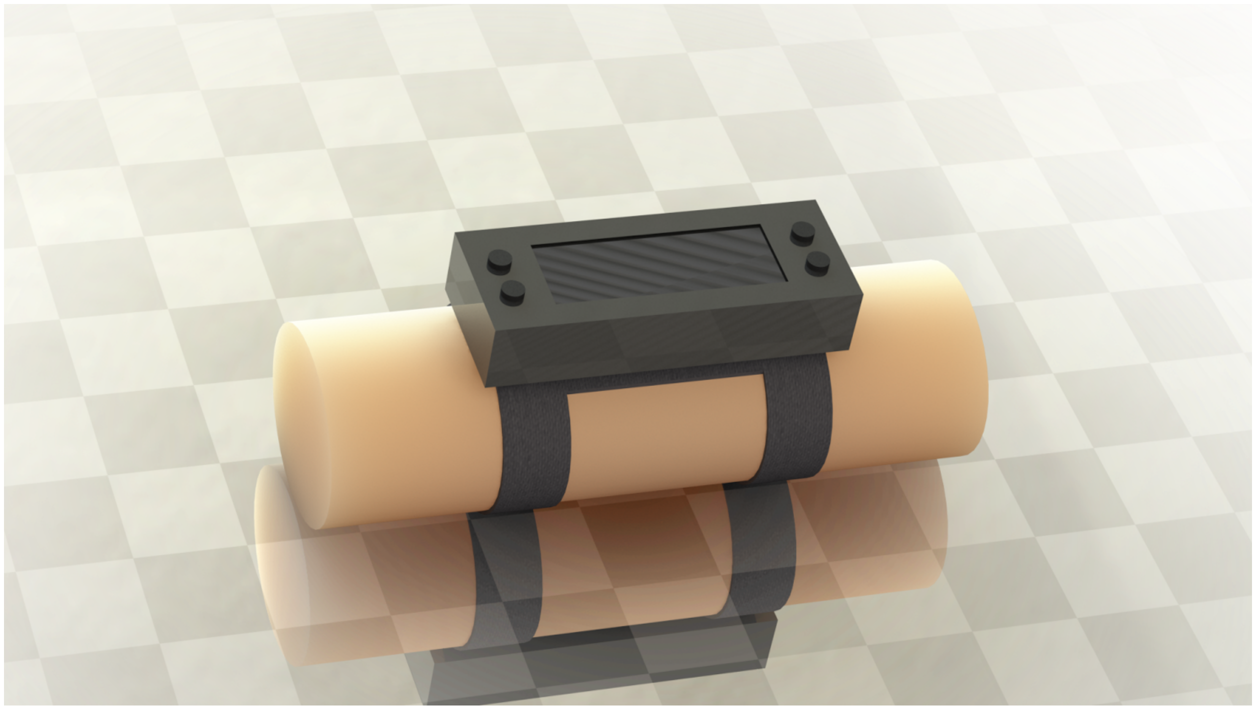


Figure 6: Housed Cyberwatch Render

To implement this new format the device needed a custom printed circuit board (PCB) that replaces all of the manual wiring that was used in the backpack design. In order to lay out the custom PCB the team utilized a program called Kicad. This allowed us to export a schematic that we in turn sent to a PCB printing company called OSH Park. The final PCB models can be seen below.

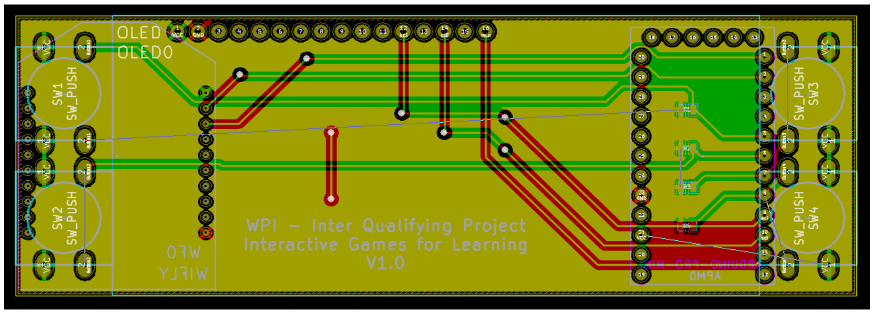


Figure 7: Design layout

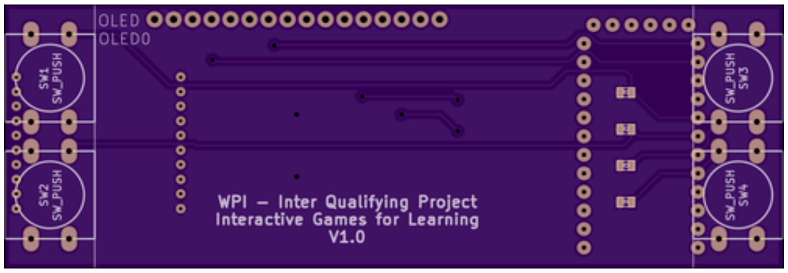

OLED
OLED0
SW1
SW_PUSH
SW2
SW_PUSH
SW_PUSH
SW3
SW_PUSH
SW4
WPI – Inter Qualifying Project
Interactive Games for Learning
V1.0

Figure 8: PCB Front

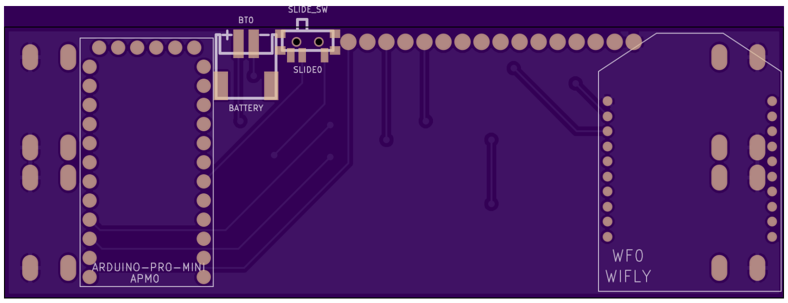


Figure 9: PCB Back

Upgrading the design and creating a PCB allowed us to replace multiple parts including the Arduino Lilypad microprocessor and power regulator. This allowed for the design to become much smaller and more manageable when working in the field as well as far more robust.

As this was our first experience with designing PCBs, it's no surprise that we ran into some issues. The PCB printing company also 'mills' the boards by drilling holes for the components. Our planning didn't take this into account, and as such we did not properly configure the holes on the PCB design. As a result, the holes were too small and parts, such as the four outer buttons, did not fit. To correct this, we crimped the leads of the push buttons in half so that contact would still be made. Additionally, the WiFly module did not fit correctly due to the same pin hole issue. We ended up leaving the WiFly module out of this prototype because it was ultimately unnecessary as we required no communication to meet our goals for the initial iteration. Finally, there were two pins for the screen that were misplaced on the board; we ran soft wires between the PCB and the screen to correct the issue.

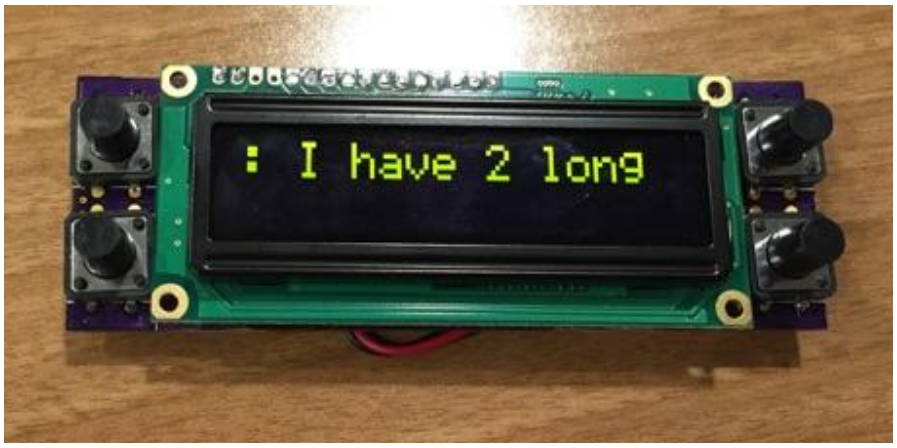


Figure 10: Cyberwatch Prototype 1 without housing

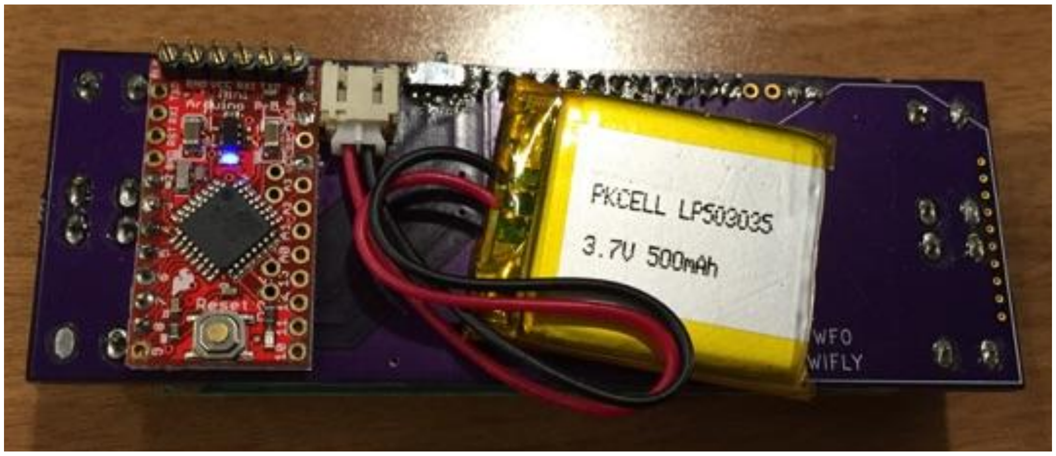


Figure 11: Cyberwatch Prototype 1 backside without housing

Eventually, we added a casing to improve the look and safety of the device and we added a wrist strap to secure the device to the user's arm. We added the casing because safety is an important factor and to prevent any damage to the watches or any harm to the user. The casing also provided the ability to make the On/Off switch for the units more difficult to access which prevented the children from accidentally or intentionally turning of their units mid game. The casing was designed from a solid acrylic box to dramatically increase the durability and secure the components. It was manufactured by laser cutting large acrylic sheets to fit the dimensions of the watch components, which were then bonded together using Methylene Chloride to form one single piece.

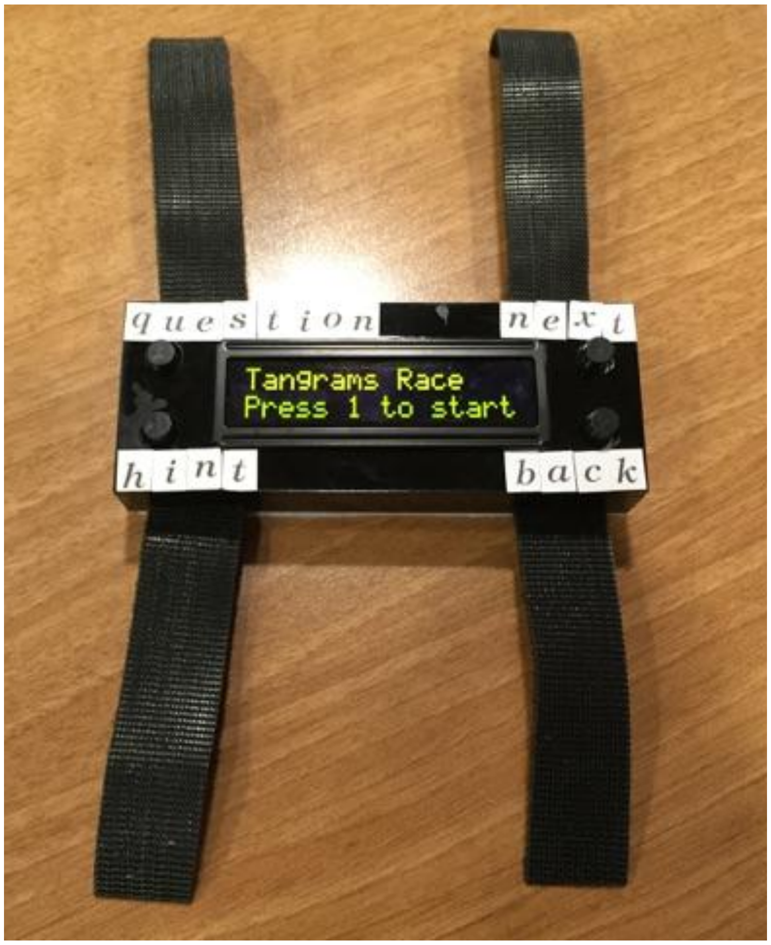


Figure 12: Housed Cyberwatch Prototype 1

3.3 Server Software

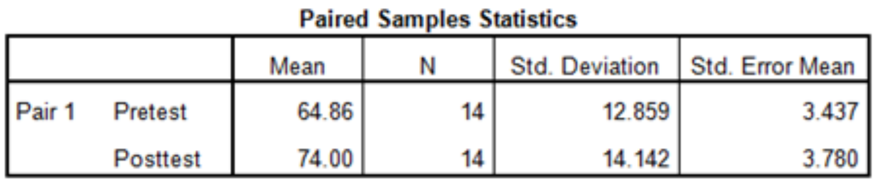
For this stage, the project didn’t require any server or other external system to run the game; however, looking ahead we began to lay out groundwork for the server that would facilitate different kinds of games. In the interest of keeping the devices robust and adaptable to a variety of games, we began designing a system that would deliver live questions and responses to the device to provide students with instructions for gameplay. The major component of this improved system is a database that stores all of the information for a game: students, their assigned devices, and the responses that should be displayed on each device. We needed to build a server that would send and receive requests to and from the devices based on each student’s progress. The original plan was to design the server using the open source TOMCAT Java Servlet and Java Server tools. We decided that this was not optimal because there are already programming languages and platforms designed to handle live communication (a la HTTP). Because of this, we began implementing a web application using the Ruby on Rails platform which simplifies web server design, is nearly infinitely scalable, and allows incredibly easy expandability through packages called gems. We'll go into more detail about our work in our description of the second prototype as the server implementation was incomplete and unused for this part of the project.

3.4 Experiment and Procedure

The first trial of the newly designed cyber-watches was performed on November 7th, 2014, with 18 children at an after school program called The Kumon Center in Shrewsbury, MA, USA. The watches were tested using the 'Tangrams Race' game, which was the M.S. project of a student in Interactive Media & Game Development (IMGD) studies (Liu, 2014). The procedure for the experiment was to quickly and informally teach brief and rudimentary geometry to the children, followed by giving them a pre-survey and mathematics pre-test (with items extracted from MCAS). Next, the children played the game, and then were given a post-survey and a similar math post-test. Math pre-tests and post-tests were counterbalanced to guarantee accurate measurement of student learning. The game was designed in a team format and each team was given a bucket of tangram pieces. The students each received a number of questions that related to one of the shapes in the buckets using mathematical terms and each question had a corresponding hint. Each team sent one student to run and grab the piece described by their current question. The students rotated through until they had collected all of the pieces, and then each team worked to solve a tangram puzzle on a table at the starting line. Each student was given a cyber-watch that was programmed with four questions, and four hints for each question. The watches were strategically handed out so that each team would collect all of the same pieces in the end. The questions were displayed on the watch screen and they could use the watch to access hints or proceed to the next question. The game was played twice with three teams of three students each.

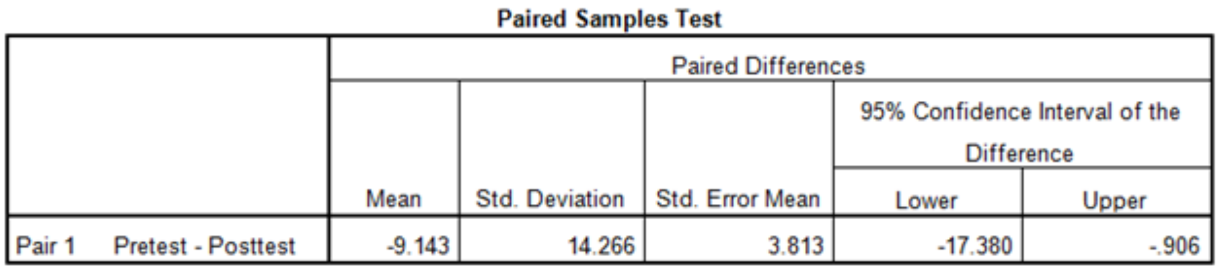
3.5 Results and Observations:

There was a ~10% difference between the pre and post-test scores, indicating the game was to some extent successful in teaching the children about geometry. You can see below that there is almost a 10pt difference in the means of the two tests.



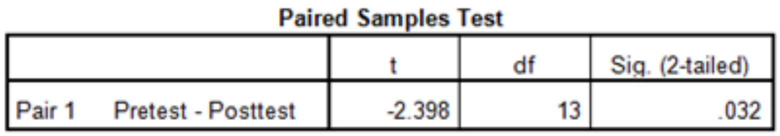
Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Pretest	64.86	14	12.859	3.437
	Posttest	74.00	14	14.142	3.780



Paired Samples Test

	Paired Differences				
				95% Confidence Interval of the Difference	
	Mean	Std. Deviation	Std. Error Mean	Lower	Upper
Pair 1 Pretest - Posttest	-9.143	14.266	3.813	-17.380	-.906



Paired Samples Test

	t	df	Sig. (2-tailed)
Pair 1 Pretest - Posttest	-2.398	13	.032

Table 1: Paired Sample Statistics for Pretest and Posttest score in the Math Tests

All of the pre and post-survey questions were based on a 1-5 scale, 1 being the lowest rating, and 5 the highest. The pre-survey focused on student's confidence and general interest in math. The post-survey focused on interest and confidence in math, as well as the student's interest and reaction to the game. The overall results showed an increase in both learning and happiness of the children playing. Figures 13 and 14 show the children playing the Tangram Race.

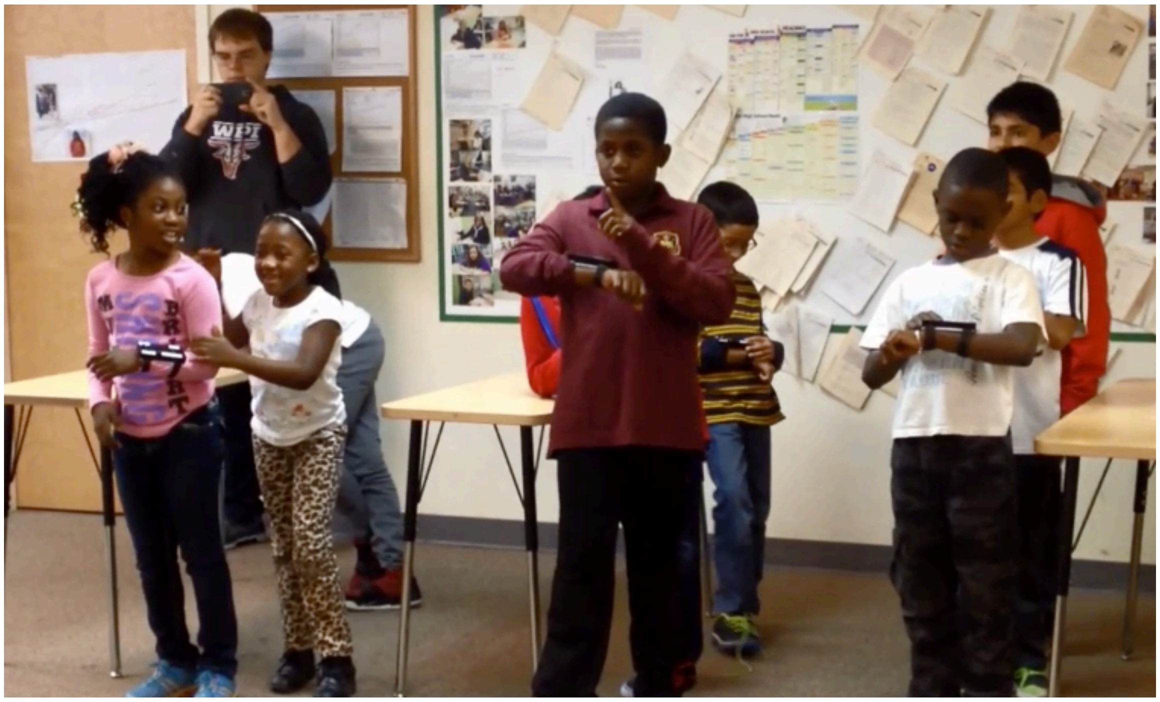


Figure 13: Pre-race team setup at The Kumon Center

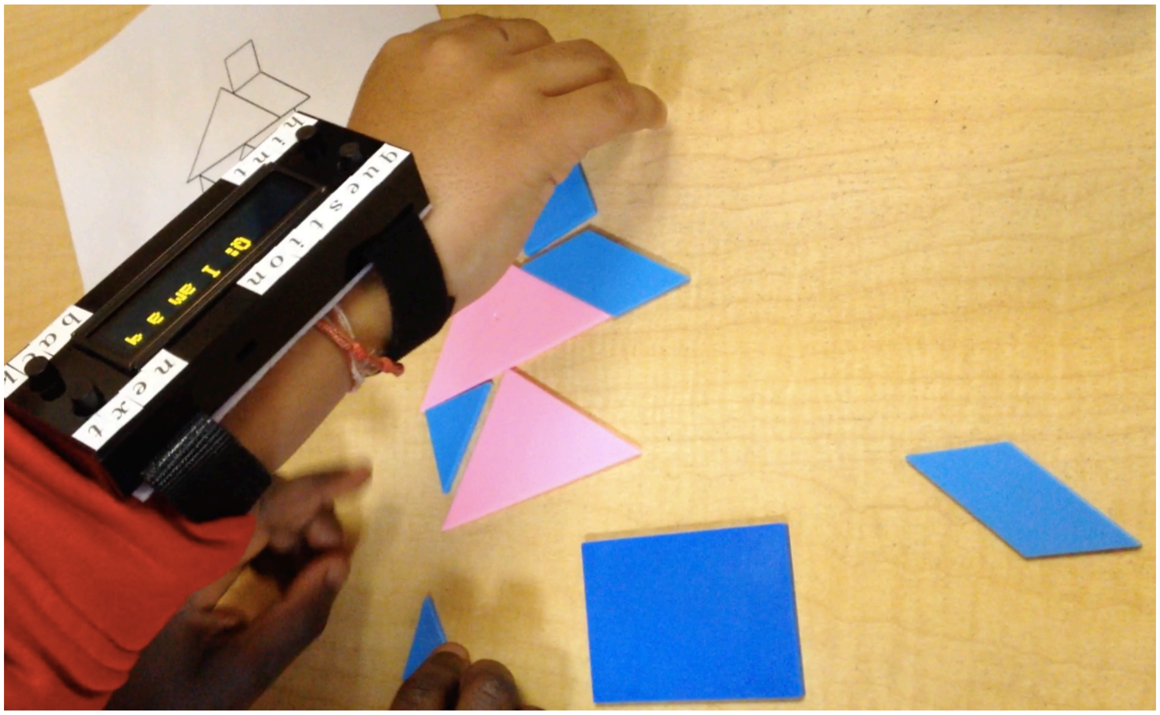


Figure 14: Mid game, building of the tangrams

Question	Average
Do you think math is an interesting subject?	4.2
In general, how confident are you when solving math problem?	4.2
How good would you be at learning something new in math?	4.2
How much do you like math?	4.5
Compared to most of your other school subjects, how good are you in math?	4.5

Table 2: Pre-survey Questions and average responses

Question	Average
After playing this game, do you think math is an interesting subject?	4.2
How confident will you be when solving a math problem in the future?	4
In the future, how good would you be at learning something new in math?	4.9
After playing this game, how much do you like math?	4.8
Do you prefer to learn math in the playground by playing games or sitting in the classroom taught by teachers?	42% classroom 58% playground
How much did you enjoy playing this game?	4
How much did you learn by playing this game?	3.8
Would you play it again?	4

Table 3. Post-survey questions and averages responses

Observing the gameplay gave us some insight into the effectiveness of the watch as well as future improvements that could be made. One issue with the watches is that the text rate is non-adjustable for the students. This means that for some students the speed with which the text scrolls across the screen was too slow, and for some the text moved too quickly. Another issue was that students have access to the hints for each question before the question has been completely displayed. We found that students were abusing this feature to immediately proceed to the hints instead of waiting to read the entire question because the hints made it much easier to find the correct piece. The last issue with the watches was that there was no indication to students of how many pieces had been collected. Consequently, students were confused because they could wrap around to questions they had already completed; an indicator for the number of pieces collected could solve this problem. One solution, although not implemented in this first prototype, would be to use RFID tags for the students to scan the pieces collected, so that the game engine is aware of the pieces collected by each team.

In regard to other aspects of the experiment, during the game there was too much outside stimulus from people and things not associated with the game and the children were easily distracted (as it was “game night” in an afterschool program). Playing the game in an actual classroom instead of in an afterschool environment would likely improve this inconvenience. Similarly, a classroom environment should improve the results of the pre and post-tests, during which we observed a few students cheating due to the seating arrangements. A classroom would provide a better test-taking environment that would give us more reliable results. Throughout the game, students were using the Tangram outline to determine pieces that would fit in the figure, so the outline should be given to each team once they have collected all of the pieces. The last concern is that students may be disheartened because of the competitive component of the game. The students on the losing team were found to have the most negative survey results in regard to the game as well as their interest in math. Providing a prize or reward for all players in the game might improve the reactions of those children and avoid students developing a negative attitude towards math and the games simply because they aren’t winning.

4. Second Prototype and Server Implementation

For the second prototype we had to modify the original design to accommodate all of the necessary components for the more complicated “Estimate It!” game, also created by an IMGD graduate student, Leigh Rountree. This meant reprinting new boards and recreating the watches with new components. This game also required the server system to be fully functional. The server was used to transfer necessary game information between itself and the devices. We also created a user interface that allows for the creation, setup, and monitoring of games.

4.1 Goals

The major goal of this second stage was to design the server and redesign the device to accommodate the Estimate It! game. We expanded the scope beyond just educational games, and began designing the interface with the idea that it could be used in a variety of fields of work.

4.2 Unit Design

In designing our second prototype we had to add several new components in order to meet the requirements of the Estimate It! game. This required us to redesign our PCBs.

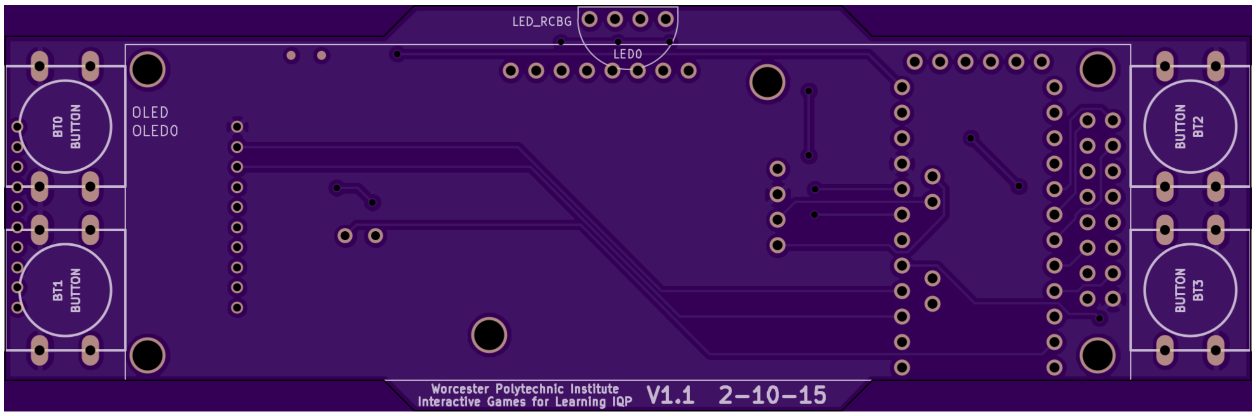


Figure 15: PCB Prototype 2, Front

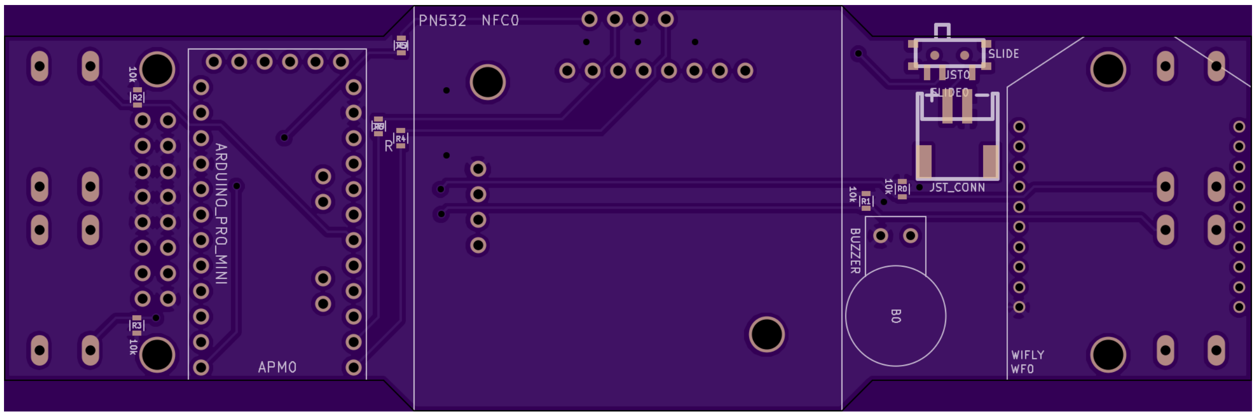


Figure 16: PCB Prototype 2, Back

We replaced the old OLED character display with a new OLED pixel display. This makes it somewhat more difficult to print things to the screen; however, the tradeoff is that we can draw anything we want to the screen, beyond a single line of text. We kept the same Arduino Pro-mini, battery, buttons, and on/off switch. In retrospect, we shouldn't have used the Arduino, and should have instead used a custom integrated circuit (IC) to accommodate the increased CPU costs of writing to the screen, as well as handling all of the communication responses. The new components were an RGB LED, a buzzer, an NFC reader, and the WiFly module. The LED and buzzer provide additional feedback to the student as they progress through a game. The WiFly module was added so that we would be able to communicate between the devices and the server. The NFC reader gives us the ability to read NFC tags which would be used to identify objects within a game. The tags could be placed on any object or a place within the game environment, and the children could scan the tag with their watch in order to progress through the game. This gives the capability to provide feedback to students on their answers such as whether or not they scanned the correct object.

For this iteration we were unable to build a plastic case for each device. While a case provides a valuable means of protection for both the device itself and the child wearing it, the ability to effectively run games has no dependence on having a securely encased device. The cases could easily be implemented at a later date.

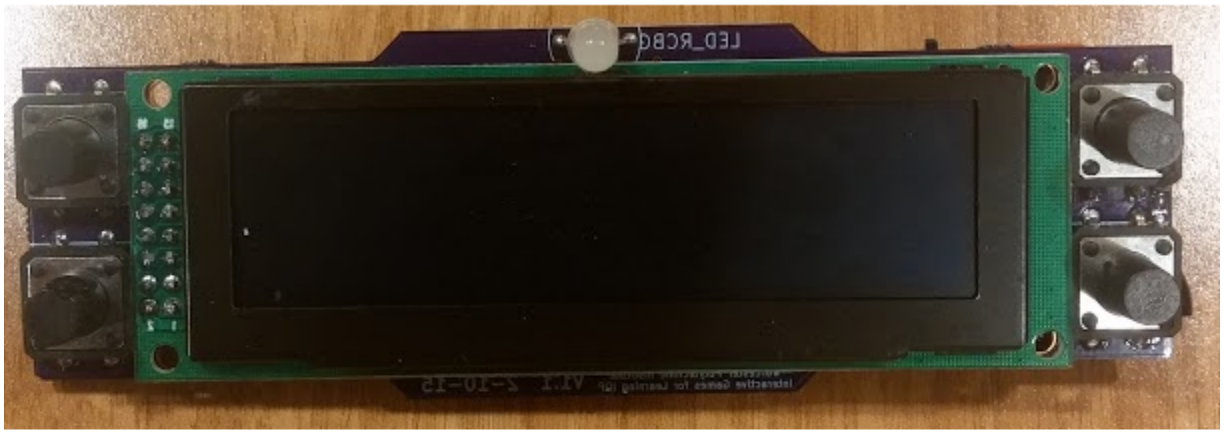


Figure 17: Prototype 2 Front

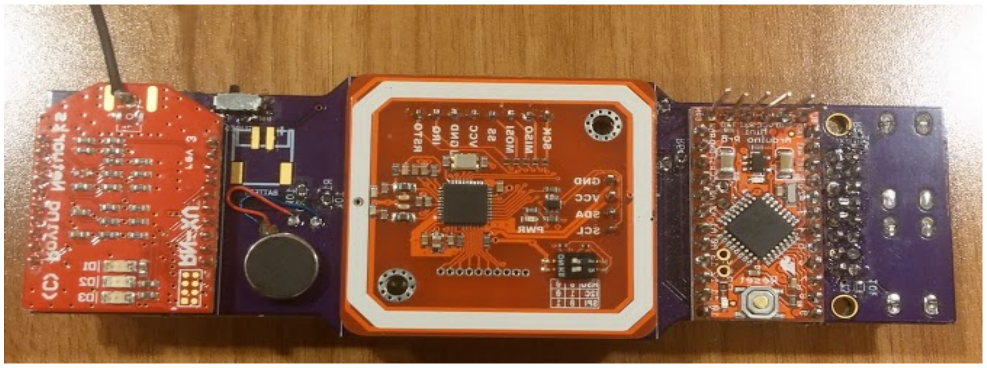


Figure 18: Prototype 2 Back

4.3 Server Software

As stated earlier, we implemented the server as a web application by using Ruby on Rails. There are seven goals the server needed to accomplish:

1. Allow teachers to create students and devices and pair them.
2. Allow teachers to create games.
3. Allow teachers to set up games.
4. Allow teachers to sign in and access games that they have previously created
5. Start and monitor games.
6. Maintain a database of all game information.
7. Send and receive responses to and from devices.

The server can be distributed into four major segments: the teacher panel, the database, the code to accommodate the server and device communication, and the device emulator.

4.3.1 Teacher Panel

The teacher panel supports the creation, setup, and monitoring of any games that a teacher has on their account. You can see examples of the teacher interface and its different sections in the figures below. The teacher panel is implemented on a Ruby on Rails server which could be run on a teacher's computer or on a remote web server. This server uses Bootstrap 3, a useful tool that assists in graphically designing a website. There are menus to access the game, student, and device lists which also hold their respective creation pages. The actual functionality for managing and creating games is split into three panels: a create game panel, a game setup panel, and a monitor game panel. In addition to the three panels, there is a landing/home page as well as the sign in and sign up pages, which follow a standard format. The graphical user interface (GUI) currently lacks any means to add teams to games or to assign a student to a team. This is because of time constraints, which we address later on. Despite this, the teams can be implemented in the database, which is described in section 4.3.2. It is worthy to note that this functionality is relatively simple to add and will likely be implemented in the future.

Currently, the home page defaults to the games menu. It lists all of the games associated with the teacher account. We'll go into more detail further along. You can see links across the top that lead to the Home (this page), Sign in, and Sign up sections. This navigation is always available. Figures 19 and 20 show the Sign in and Sign up pages. These pages are very similar to the account creation pages of other services. In the future, we may decide against any need for teachers to maintain accounts, but for the time being, the functionality remains.

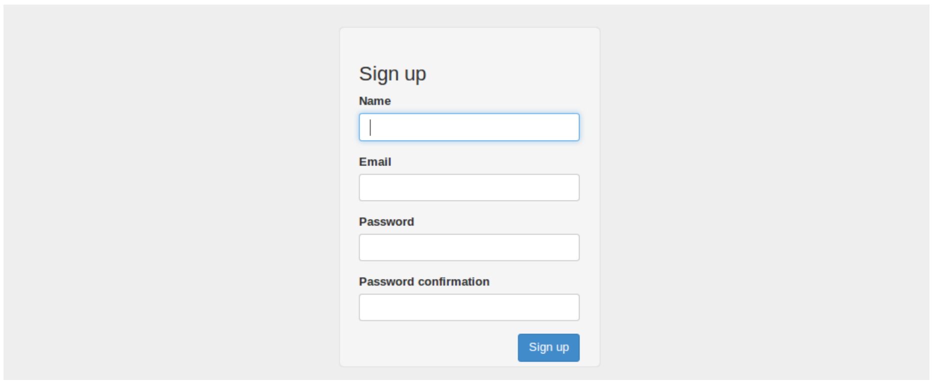


Figure 19: Sign up page

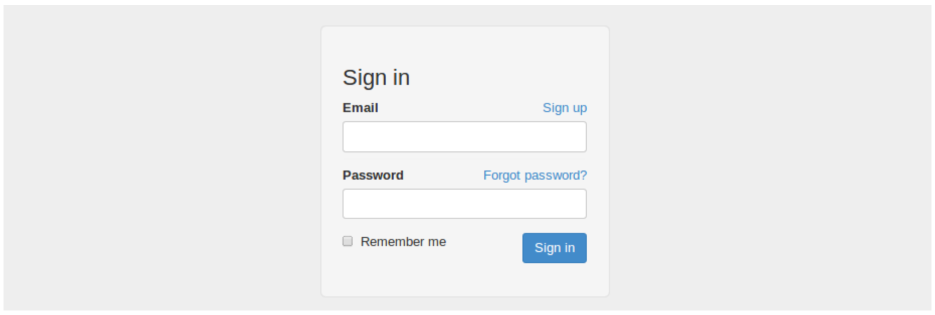


Figure 20: Sign in page

This next section displays and describes the different menus. The teacher panel will be explained in further detail later. There are three menus that list the Games, Students, and Devices, all of which are accessible from the top menu-bar of the web page.

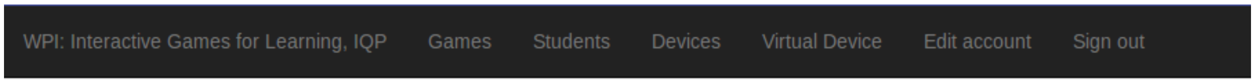
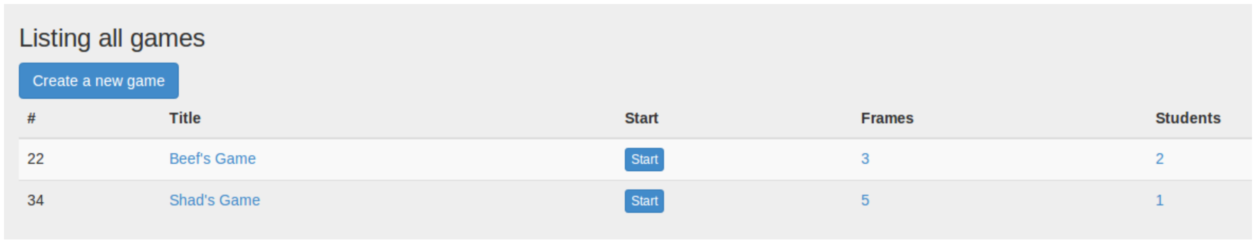


Figure 21: Menu-Bar

Selecting the Games menu displays the page illustrated in Figure 22, which lists all of the currently available games on the account that is currently signed in. The game id, title, number of responses (frames) and number of students is listed for each game. From here a teacher can begin creating a new game, edit an existing game, or start any games that are available on the account. The create game will navigate to the create game panel, which is better explained later on.

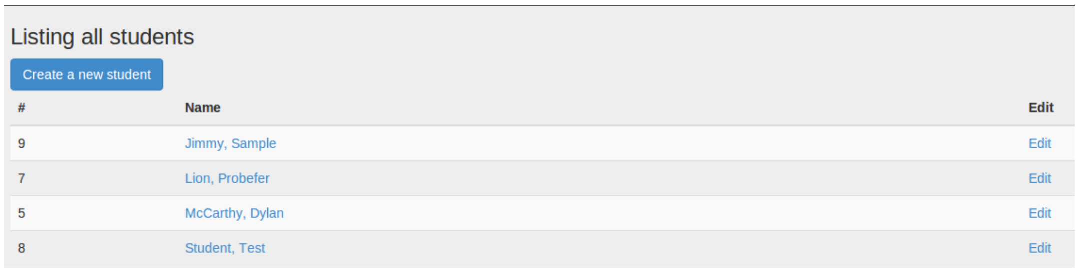


Listing all games

#	Title	Start	Frames	Students
22	Beef's Game	Start	3	2
34	Shad's Game	Start	5	1

Figure 22: Games Menu

Selecting the Students Menu displays the following page which lists all of the students that a teacher has added. Teachers can edit any of the students they already have, or create a new student, which brings up the page shown in Figure 24.



Listing all students

#	Name	Edit
9	Jimmy, Sample	Edit
7	Lion, Probefer	Edit
5	McCarthy, Dylan	Edit
8	Student, Test	Edit

Figure 23: Student Menu

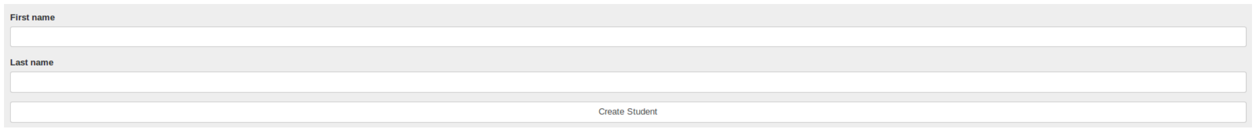
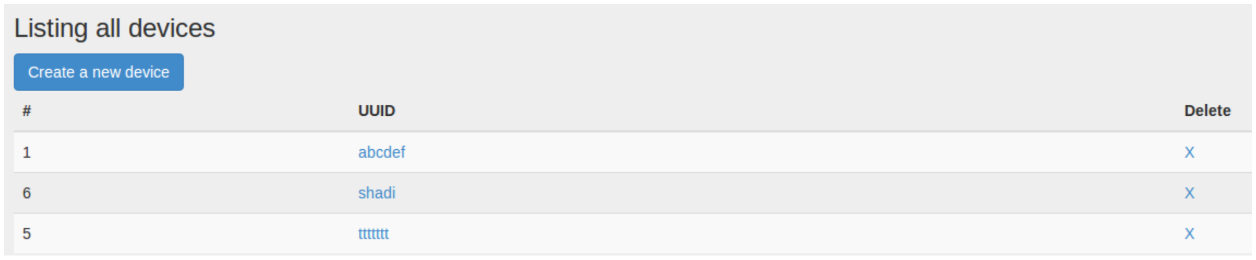


Figure 24: Create Student Page

Selecting the Devices Menu lists all of the currently registered devices and their universally unique identifiers (UUIDs). From here a teacher can create a new device or delete any of the current devices.



Listing all devices

#	UUID	Delete
1	abcdef	X
6	shadi	X
5	tttttt	X

Figure 25: Devices Menu

Now we get into the Teacher Panel which takes us through the entire game process. We create a game, set it up, and monitor games through the panel. The create game functionality is accessible from the Games Menu by using the Create Game button on the page.

The Create Game Panel is where we define the progression of the game. Responses are added with the specified content and potential inputs that a student can make from the device. First a user defines a title for the game. After that, responses are added one by one, by first selecting the type of response to add. There are three possible response types: Text, LED, and Buzz. As the response type is selected, the content input changes accordingly so the user will enter text for text, a color for the LED, etc. A text response would represent any kind of text to be printed to the display. This could be a question, a hint, or simply a welcome screen. The LED response triggers the LED to display the selected color. When the LED response type is selected, a color selector becomes available for the user to easily set the LED color. For the buzzer, the user simply enters the amount of time they want the device to buzz for in milliseconds. In order to specify the next step of the game, the teacher will first press the circular plus button in the lower right-hand corner of the page. Next, they will change the “Do Nothing” dropdown to the correct response to jump to if that button is pressed. For example, the first step could be of type text saying “Welcome to the Game”, with every button dropdown set to “Go to Step 2”. Step two could be of type text with further game instructions. In this example, the cyber-watch would first display the welcome message, and when any button is pressed, display the second step with further game instructions. The teams menu at the top allows the teacher to specify the number of teams and the number of players per team. This in turn will generate more “Do Nothing” dropdowns, with one set of four dropdowns (for each of the four buttons) per team. Building the game in this dynamic manner allows us to specify different steps for different teams, and allows us to allocate specific steps to specific teams. Using this interface, nearly any game can be designed as long as it follows the state-based structure of a finite-state machine.

Upon entrance to a state, a light might turn on, or text can be displayed. Transitions between states happen instantaneously, via timers, button presses, or scans using the NFC scanner.

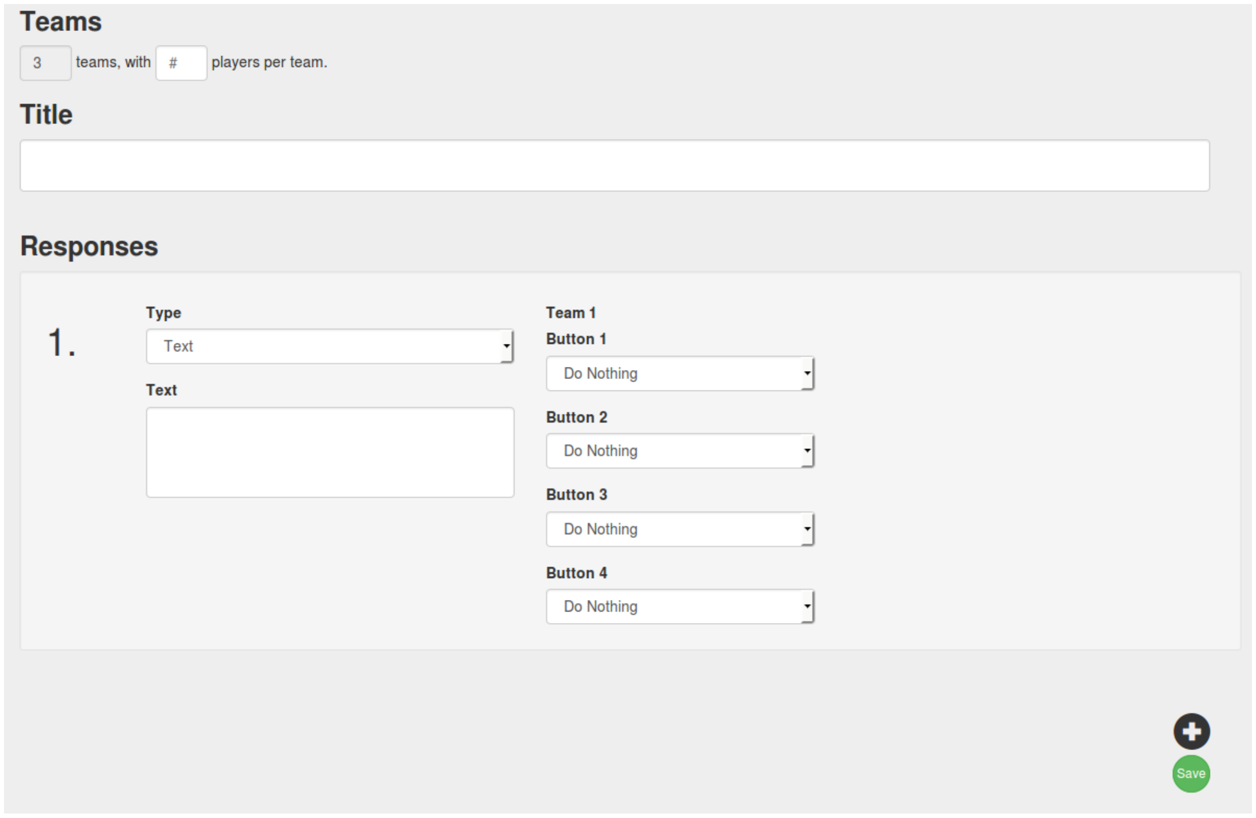
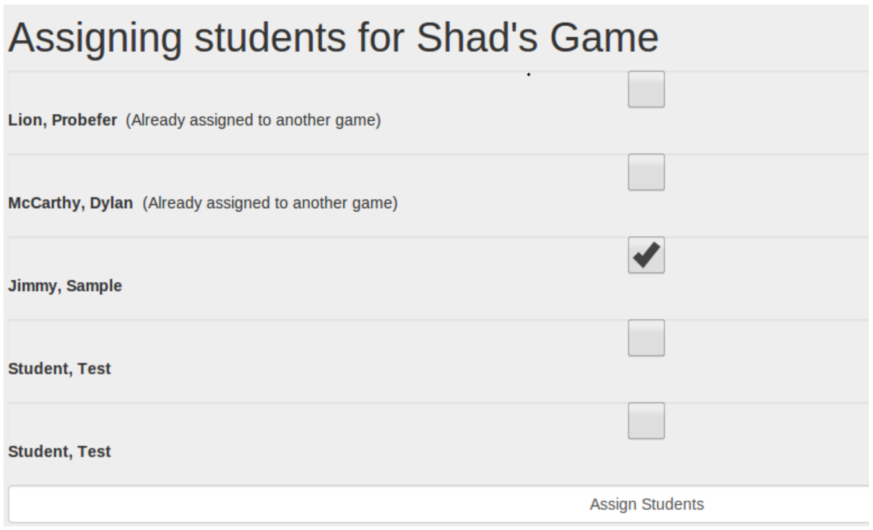


Figure 26: Creating new game menu

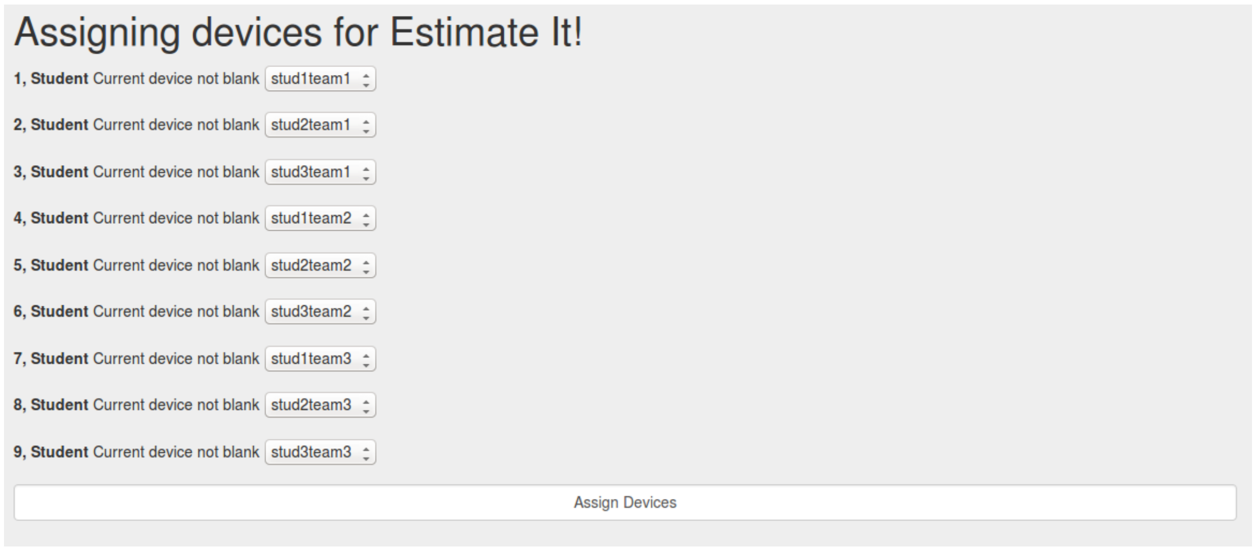
The next stage is the student association page of the game setup panel. The game setup is where a teacher selects a game and can assign students to the game and assign devices to each student. Figure 27 shows the student association page. All of the students that a teacher has entered are displayed and the teacher simply checks off which students to add to the specific game. It also alerts the teacher to which students already belong to a game. If checked, these students will automatically be re-assigned to this new game.


Assigning students for Shad's Game

Lion, Probefer (Already assigned to another game)	☐
McCarthy, Dylan (Already assigned to another game)	☐
Jimmy, Sample	✓
Student, Test	☐
Student, Test	☐

Figure 27: Assigning players to a created game

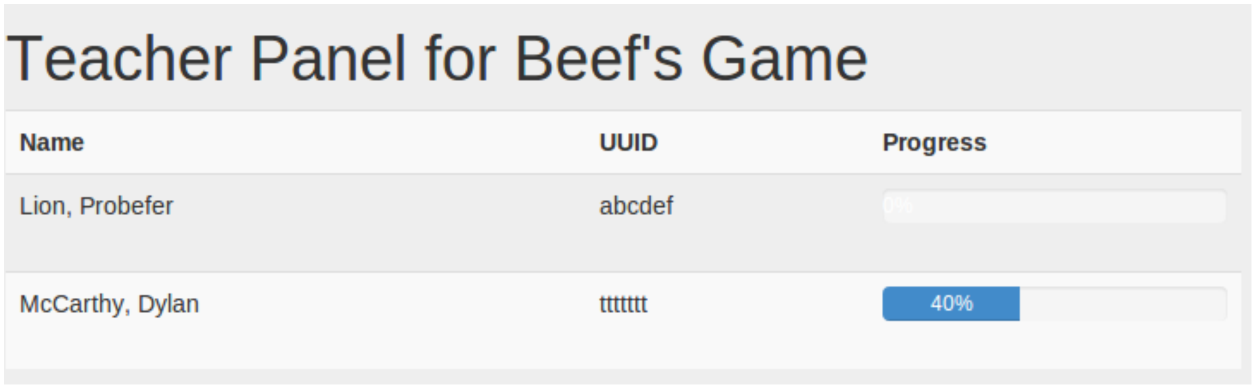
Next, there is the device assignment page of the game setup panel, shown below in Figure 28. This page displays each of the previously checked students and allows them to be associated with a previously inputted device via a drop down menu.


Assigning devices for Estimate It!

1, Student Current device not blank stud1team1
2, Student Current device not blank stud2team1
3, Student Current device not blank stud3team1
4, Student Current device not blank stud1team2
5, Student Current device not blank stud2team2
6, Student Current device not blank stud3team2
7, Student Current device not blank stud1team3
8, Student Current device not blank stud2team3
9, Student Current device not blank stud3team3

Figure 28: Device Assignment Page

Finally, there is the panel to monitor a game that has already been started. There is currently no implementation for adding teams to a game, so there is no information being displayed on team progress in this panel. The only information is the progress of an individual student. As you can see below, for each student we display the UUID of their assigned device and the progress of that student, which is based on the responses that they have progressed through.



Teacher Panel for Beef's Game

Name	UUID	Progress
Lion, Probefer	abcdef	0%
McCarthy, Dylan	ttttttt	40%

Figure 29: Teacher panel showing student progression

4.3.2 Database

The database is designed using the idea of a list of responses associated with each game. The response table represents a step by step progression of the game, where each device action directs to some other response in the table. The idea is that this opens up the game development to be as versatile as possible. There are 6 tables in the database, however for the following graphic and information we are going to ignore the user table. The user table simply stores account information for the teacher and it is entirely independent of successfully running a game, so for simplicity we will exclude it. Below is an example of the relations between the tables for the current implementation of the database. Each block represents one of the tables, and contains the tables' title and all of its columns. The arrows and lines show the relationship between the identifiers and the table entry they would direct to.

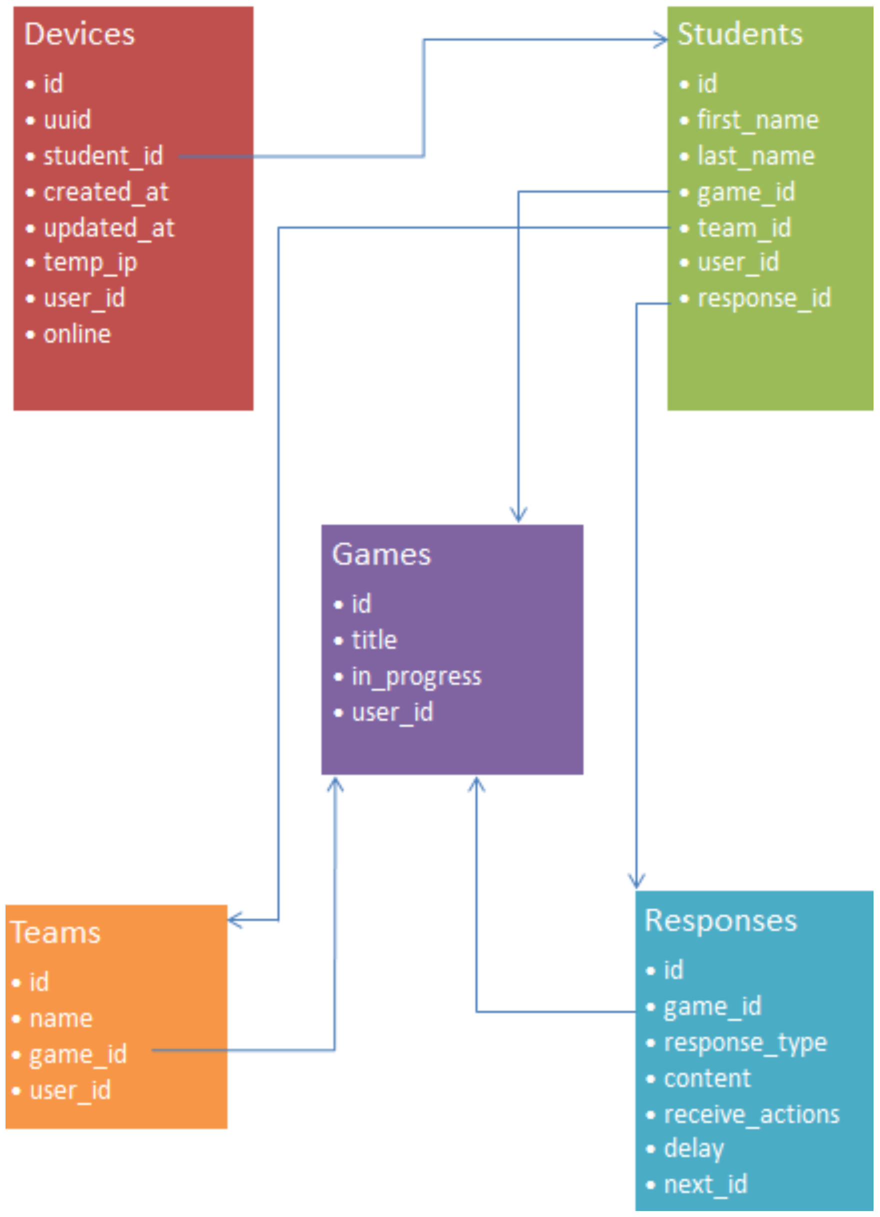


Figure 30: Database Relations

Below, there is a description of each table and its fields to assist in understanding the database structure. All of the tables maintain timestamps to represent the time an entry was created and the last time an entry was updated.

devices: **id, uuid, student_id, created_at, updated_at, temp_ip, user_id, online**

The **id** is a primary key.

The **uuid** is an assigned device-specific unique identifier.

The **student_id** creates the relation between the device and the student assigned to that device.

The **temp_ip** is used to keep track of the IP address that the device is using to connect to the server.

The **user_id** keeps track of the teacher account that the device is associated with.

The **online** boolean is used to tell the server if a device is currently active or not.

students: id, first_name, last_name, game_id, team_id, user_id, response_id, player_number

The **id** is a primary key.

The **first_name** field is the text of the student's first name.

The **last_name** field is the text of the student's last name.

The **game_id** field links the game that the student is currently playing.

The **team_id** represents the team that a current student is assigned to.

The **user_id** keeps track of the account that the student is associated with.

The **response_id** represents the response in the response table that a student is currently on..

The **player_number** represents the order of the player on the team. For a three person team, the player numbers would be 1, 2, and 3.

games: **id, title, in_progress, user_id**

The **id** is a primary key.

The **title** is simple text for the name of the game.

The **in_progress** field is a boolean to determine if the instance of the game is active or inactive.

The **user_id** keeps track of the teacher account that the game is associated with.

responses: **id, game_id, response_type, content, receive_actions, delay, next_id, stall**

The **id** is a primary key.

The **game_id** defines the game that the response is associated with.

The **response_type** defines the type of the response. This could be text, buzzer, or LED.

The **content** defines the actual content describes the information for the associated **response_type**.

The **receive_actions** is a hashmap that defines the net response to go to depending on the read input.

The **delay** represents the time for some response to be performed. Used to define the times the LED or buzzer are activated.

The **next_id** defines what the next response will be after this current response has finished.

The **stall** boolean indicates whether or not to stop all players on a step until the entire team has reached that step.

teams: **id, name, game_id, user_id, team_number**

The **id** is a primary key.

The **name** represents the name of a team.

The **game_id** represents the game that a team is associated with.

The **user_id** keeps track of the teacher account that the team is associated with.

The **team_number** keeps track of the order of each team.

4.3.3 Server and Device Communication

The communication between the device and the server are accomplished by using websockets. When a watch is turned on it opens a web socket to the server performing a websocket handshake to establish a connection. Once connected, the watch immediately sends an authentication request to the server with its UUID. This is our own authentication request, separate from HTTP authentication. Once the device has been authenticated, the server will update the device's entry in the devices table in the database to specify that this device is online. Once the device is online, the socket connection remains open but we send no messages between the device and server until the game the device is associated with begins. Note that the websocket protocol implementation we utilized has a "ping", "pong" method of keeping the socket open. Every specified amount of time (usually a second or so), the server sends the device a special "ping" message. If the device does not reply with a special "pong" message, both devices assume the connection has been closed and can take action accordingly.

On game start, the game queries the database for all of the students associated with it, and looks for all those students that have a device that is currently online. Once the list of online devices has been established we immediately push the first response associated with the game to all of the devices. The game functionality continues through the use of seven main functions within the game.rb model: gameStart, transmitResponse, reportToPanel, resetIfDone, returnNextResponse, gameEnd and interpretNextAction.

gameStart(student_id): This method takes a student and determines the title of the current game associated with the student. It then transmits the title of the game to the device.

transmitResponse(student_id): This method takes a student_id and determines the current response associated with that student by looking at that student's response_id in the students table. The function reads the response from the table, transmits it to the device, and increments the response_id appropriately. The function recursively calls itself until it reaches a response that depends on user input, because it will just continue incrementing the response and proceeding each time.

reportToPanel(student_id): This function takes a student_id and determines what response the student is currently on by consulting the students table. The function then determines the total number of responses in the game, which it compares to the student's current response to determine the total progress the student has made. Note that with the current team implementation, some students are never meant to see certain responses. Due to this, the progress percentage is not accurate, because a student's progress is measured against the entire list of responses, instead of just the responses they are intended to progress through. In the future the progress calculation should be modified to more accurately determine a student's progress.

resetIfDone(tp_uuid): This function determines if all of the students are at the end of a game, and will reset the game to no longer be in progress, and will reset all students to a clean state(no longer assigned to a game).

returnNextResponse(response_id): This method takes a response_id and returns the next response in the progression.

gameEnd(student_id): This method sends a message to the devices to inform them that a game has been completed.

interpretNextAction(uuid, action_id): This method is called whenever there is user input from the watch. This could be either a button press or an NFC tag read. When input is received, the method determines the response that is expecting the received input by finding the current response of the student associated with the device uuid. The action_id is used with the receive_actions hashmap in the responses table to determine the id of the next response based on the input, and then updates the student's response_id. The function then calls the transmitResponse function with the student_id to return a response to the device.

There are methods to call these helper functions through the web-socket connection, which are: client_connected, client_disconnected, button_pressed, and nfc_scanned. As the inputs occur on the devices the appropriate function is called, which then refers to the helper functions on the server. These functions are located in the device_controller.rb file.

Besides creating the functions to facilitate the server-device communication, we needed to create our own format for the responses being sent to the device. For each response we specify the game_id, the response_type (text, led, buzz), the content of the response, and the receive_actions (the next response based on device input). Additionally, the stall condition can be specified which is used to halt a team until all players arrive at that response. Leaving the stall field out of a response defaults the stall state to false. Below is an example response from the seeds.rb file we used to populate our database.

```
Response.create(
        game_id: game.id,
        response_type: "text",
        content: "{:text => 'That was not the right color...\nPress any button to try again.'}",
        receive_actions: "{'button1' => 7, 'button2' => 7, 'button3' => 7, 'button4' => 7}");
```

Due to time constraints we were unable to build the fully functioning GUI interface required to specify the Estimate It! game completely. We created the responses using the syntax above, and added them to the game using the seeds.rb file.

The Estimate IT! Game was specified by Leigh Rountree (IMGD Masters student) and consisted of different levels, and students working in teams. Watches are synchronized for people in the team at all times, so that they cannot move on beyond certain moments/states unless every person in the team has gotten there.

4.3.4 Device Emulator

To facilitate our testing we developed a device emulator on the server. Using JavaScript, we created an emulator to fully mimic the actions of an actual device. It has functionality to represent each of the different components on the physical watch. The emulator uses the same communication methods as the actual device does. Just like the physical devices we designed, the virtual device must be assigned to a student (by assigning it a UUID), and can only receive messages from a game that has begun. The top circle represents the LED. To recreate NFC tag reads, NFC tags were supplemented with specific sequences of button presses.

To use the virtual device, you must first enter a UUID that is assigned to a student associated with a game that has begun. After the UUID is entered and set, the virtual device can be powered on by pressing the switch displayed in the bottom left. At this point a user will receive responses and can progress through the game just as they would with a physical device.

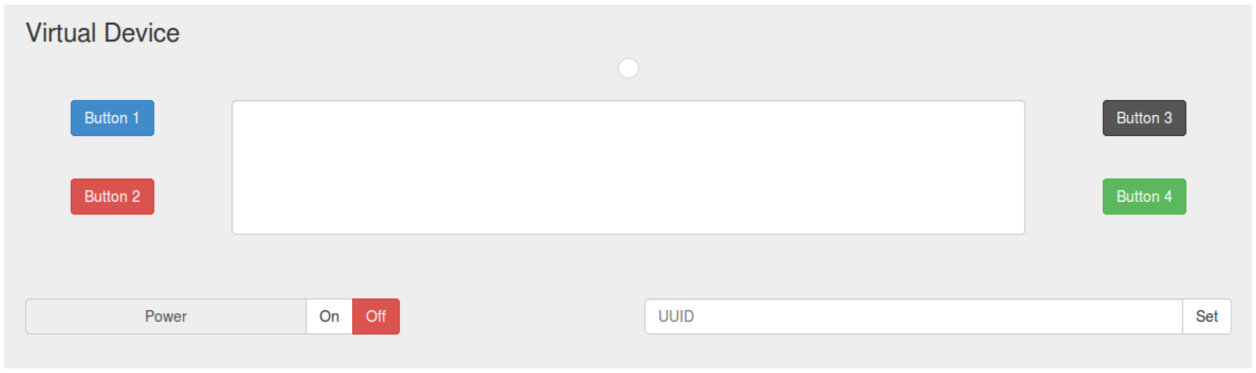


Figure 31: Virtual Device Emulator

The device emulator proved to be more than just a means of testing our server implementation. Due to the constraints of the Arduino Pro-mini used in our smart-watches, we were unable to implement all of the requirements for the Estimate It! game on our physical devices. The costs of writing to the screen, processing device inputs, and maintaining a wireless communication proved to be too much for the processor we built our device around. It was infeasible to redesign and reprint PCBs around a processor that met our requirements due to time constraints. In order to trial the Estimate It! game with a group of students we resorted to using the virtual device in place of the physical devices which were not capable of facilitating all of the games requirements.

4.4 Playing Estimate It!

Despite issues with our device, and in getting the game creation pages to be able to facilitate all aspects of the Estimate It! game we were able to trial the game with a group of students. The game was built by manually entering messages, teams, students, and state specifications and state transitions constraints as well as device specifications into the database via a seeds.rb file (an initialization file that populates the PostGres database completely, and sets up initialization parameters). Once the database was properly seeded we were able to launch a game from our web page, and monitor individual progress via the game monitor panel.

The Estimate It! game was designed as a scavenger hunt that is meant to encourage practice and learning of a child's estimation skills and properties of geometric objects. As they play, students are delivered responses that provide clues (a mathematical description) about some 3D shape (volumes such as spheres, cubes, prisms and cylinders) that is hidden in the play area. Each question is paired with a "hint" that a student can access by pressing the specified button, that is meant to teach students as they get stuck and cannot find the object. Students use their knowledge of geometry and estimation skills to determine what shape they are being asked to find.

At the same time, each shape is assigned some sequence of button presses (represented by the button colors) that the student must enter into the virtual device to assess whether that is the correct sought object. As the sequence is input, the students receive immediate feedback on whether or not the button press was correct. There are 3 levels to the Estimate It! game. Level 1 calls for the students to find different objects as individuals, where the accomplishment of each student makes the whole team win and progress further (we call this cooperation, as players are given individual tasks that are compatible for the team to succeed). Levels 2 and 3 require the teams to work together to find single objects (we call this collaboration, as all students in the team discuss about the sought object).

Synchronization of watches for a team happen through a "stall" condition, so that students in the same team are stalled at specific states in the game, so that the team doesn't progress until all members have successfully input the sequence of the correct object, or managed to find the right object. The first level of the Estimate It! Game is specified in Figure 32.

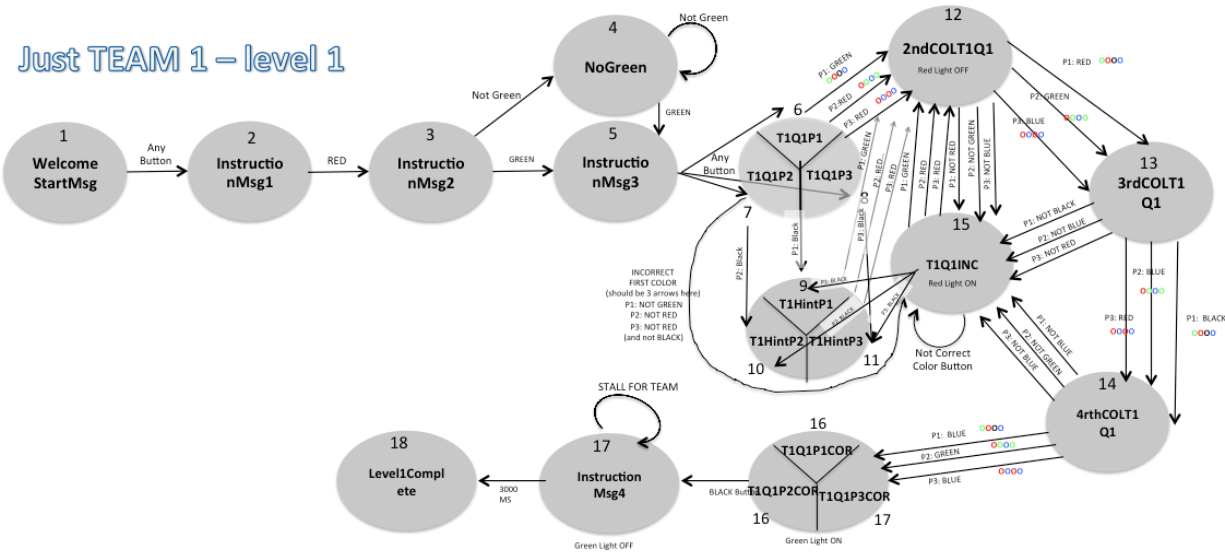


Figure 32. Finite State Transition Diagram for Level 1 of the Estimate it! Game

Figure 32 shows the level of complexity that the software component (server) needs to handle. As students move through states during the game, students within the same team need to be synchronized to some extent, as well as their watches. For instance, as students push buttons and read messages and arrive at state 6, after *InstructionMsg3* (a label that corresponds to a full message in two lines on each students' watch), the members of the team diverge into 3 different messages, as every member of the team will start searching for a different object. As students search and find objects that follow geometric and measurement constraints (with a 12 inch dowel provided as the only measurement tool), they try answers (which they input as a sequence of colored buttons) , request the watch for help/hints, until they get to the correct object. When a student reaches state 17 (and read *InstructionMsg4*) they stall and cannot move on until all members of the team have reached that point when they found the correct object. The role of the children is now to help other members of their team to find the objects that they are looking for.

The specification of a game via a state transition diagram, with events to transition between them, and being able to enter them (author games at this level of detail) into the system, for later transmission via WiFy to the watches, as well as monitoring of students and teams, makes this a sophisticated piece of machinery, infrastructure that can serve a variety of educational games (or simply games) for different disciplines and children ages.

We tested the Estimate It! game with a group of thirteen 4th grade students at the Kumon Center on May 1st, 2015. The study was a pilot study, and we divided the 13 children to play two separate games, the first using two teams of three players and the second using two teams of two players and a team of three players. There were a number of issues we encountered during our trial at the Kumon Center. For reasons unknown, the machine we used to run the server failed to stall students correctly. This issue was overcome by manually editing the database to place students at the correct response which allowed them to continue gameplay. Another issue was that the students were repeatedly being disconnected from the game. There were several reasons for the disconnects: students accidently turning their virtual device off, students accidently leaving the webpage, or the device disconnecting itself from the network. This issue can be solved by using the physical device instead of using smart phones or tablets to run the game through the virtual device.

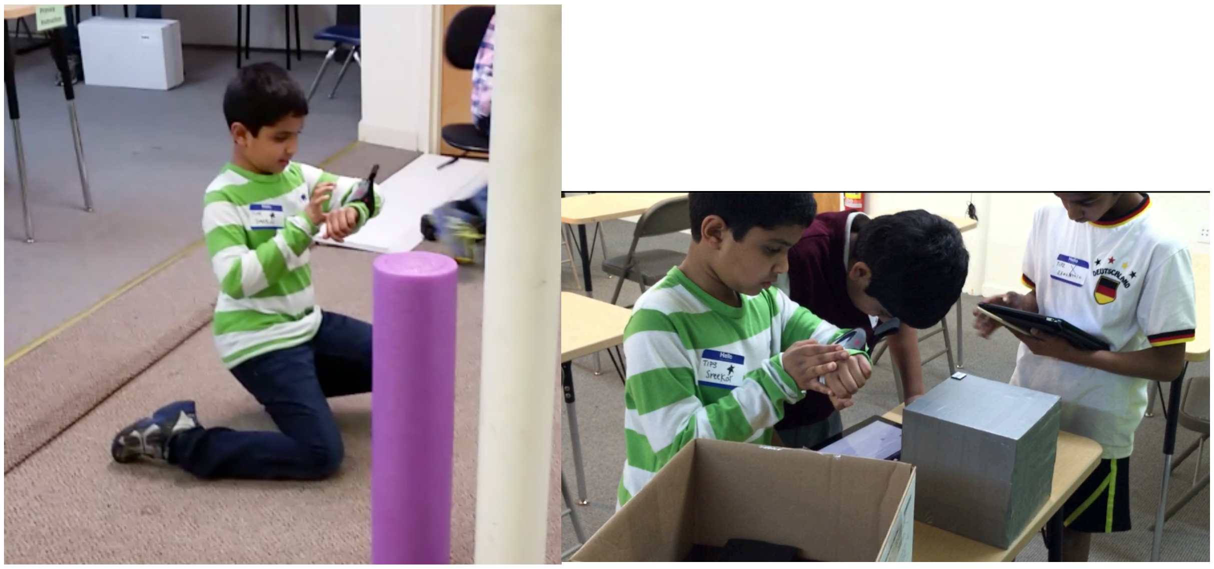


Figure 33. Student analyzing the Cylinder object and verifying that it satisfies constraints on his Cyberwatch (left); Full group analyzing a cube and verifying if this one satisfies constraints (right)

In spite of these technical issues, we were able to complete the two instances of the Estimate It! Game (see Figure 33 for a team of students playing the game), and the IMGD graduate student is analyzing results in relation to student learning and motivation. We (IQPs) informally wrote observations by following teams, and analyzed students reactions, which were generally positive. Students enjoyed playing the game, though they found the questions and hints to be too simple, and reported they could have used more challenge.

Students also commented on the devices themselves: all students would have preferred to use tablets instead of smart phones for the larger display; also, the majority of students would have preferred tactile buttons instead of touch screens. Students seemed satisfied after the game; regardless of how well they performed, which was encouraging, as the technology and Internet connection did not fail enough to make them unmotivated. We believe that in the future we can obtain much more meaningful results by running the game via dedicated devices, and a server that is fully operational.

5. Future Plans

In our final design stages we started working towards a versatile implementation that we believed would facilitate a variety of games. By changing our whole design to use a set of responses, a step-by-step progression for a game, we designed a system that should in theory allow for the creation of any kind of game. The games have the potential to use any of the current components: pixel display, buzzer, and RGB LED. Based on the current server design it should be relatively simple to add new components that would simply constitute different types of responses. There is a lot of potential for our system to accommodate all kinds of learning games that could benefit from the use of a technical device.

Currently the server is not fully implemented. The GUI for creating a game needs to be redesigned such that it can more elegantly handle creating responses that are specific to an individual or team. We believe that using a wizard to facilitate game creation would make the implementation easier to build upon, more visually appealing, and easier to use. Necessary additions to the front end include the ability to edit a game and improvements to the game monitor panel to provide more accurate information about games and players. Furthermore, the backend should be refactored to improve efficiency. A function should be created that can traverse the game tree; this would allow the system to better handle the delivery of responses to students, and would allow for easier, more accurate monitoring.

The device itself requires a complete overhaul. The PCBs need to be redesigned and reprinted, ideally with an IC that would be significantly more powerful than the current Arduino processor. Improving the device processor will remove the necessity to do any extreme code optimization. This should also make the device thinner, and a smaller device is more suitable for young kids. The new device should accommodate all of the components on the current prototype, as well as the NFC reader which went unused. Similar to the first prototype, a final design should implement a case and a strap.

6. Conclusion

This project went through much iteration to reach the current state, and could undergo many more iterations in order to become a polished and finished product. However, we have put countless hours into building what has culminated in a mobile platform that can serve specialized data from a server, allow games creation and game play in real-time.

Although we chose to focus on the educational aspect of this platform, the potential for expansion in nearly any field such as business optimization, quality control, or firefighting is unlimited. Smartphones are popular and perform many functions well but they are prohibitively expensive. Instead, the device we have created is very inexpensive and performs one function without fail. Furthermore, the networked nature of this device makes it a prime candidate for future development as the industry of tech wearables grows at an ever increasing rate.

We found success with this platform when applied in an educational setting, as we used it to engage young students' attention and encourage teamwork, creativity, and problem solving. Ultimately, this platform was built on the concept of scalability and with the idea of having a dynamic nature so as to be able to adapt to any game type. With the assistance of Professor Ivon Arroyo, Leigh Rountree, and many other mentors and helpers, we were able to build what we are proud to call a new technology – that is, we were able to build a system from the ground up, applying our knowledge gained here at Worcester Polytechnic Institute. This, we believe, is telling not only of the success of the IQP but also that the knowledge gained at WPI will assist us in our future endeavors.

References

Alibali, M. W.; Nathan, M.I J. (2012): "Embodiment in Mathematics Teaching and Learning: Evidence From Learners' and Teachers' Gestures", Journal of the Learning Sciences, 21:2, 247-286 https://docs.google.com/file/d/0B_dUxqCpzSjOSVhZblVGRDlwYlk/edit

Arroyo, I. (2014) "NSF Cyberlearning: Grant Proposal on Cyber Hoodies" https://docs.google.com/file/d/0B_dUxqCpzSjOeEpFSGI5cHhpdEU/edit

Liu, Y. (2014) *Tangram Race* Mathematical Game: Combining Wearable Technology and Traditional Games for Enhancing Mathematics Learning

Thesis presented towards completion of the MS in Interactive Media and Game Development at WPI.

Lui, M.; Slotta, J. D. "Immersive simulations for smart classrooms: Exploring evolutionary concepts in secondary science" *Ontario Institute for Studies in Education, University of Toronto, Toronto, Canada*; n. p. *Google Drive.* Web. https://docs.google.com/document/d/1S4OklqYS5lWAn2D118eUB4hskUbneNVKaOKEnOH8RLo/edit#heading=h.gjdgxs

Moreau, D. "Constraining movement alters the recruitment of motor processes in mental rotation" Springer-Verlag Berlin Heidelberg 2012, https://docs.google.com/file/d/0B_dUxqCpzSjOWG5fU1VtLWI1ZHM/edit

Wilson, M. "Six views of Embodied Cognition" (2002): n. p. *Google Drive*. Web. https://docs.google.com/file/d/0Bx06nt2HVJ-FTC1TcDhSQzhCUEk/edit

Woolf, B.P. (2010). A Roadmap for Education Technology. funded by the National Science Foundation # 0637190. http://www.cra.org/ccc/files/docs/groe/GROE%20Roadmap%20for%20Education%20Technology%20Final%20Report.pdf

Zualkerman, I. A. "ZUL – A Light-Weight Architecture for Zigbee-based Ubiquitous Applications." (2011): n. p. *Google Drive*. Web. <https://docs.google.com/file/d/0Bx06nt2HVJ-FYzlIMHJNMnY0QWs

7. Abstract

7.1 bounce.cpp

```
#include "Arduino.h"
#include "bounce.h"

#define DEBOUNCED_STATE 0
#define UNSTABLE_STATE  1
#define STATE_CHANGED   3

Bounce::Bounce()
    : previous_millis(0)
    , interval_millis(5)
    , state(0)
    , pin(0)
{}

void Bounce::attach(int pin) {
    this->pin = pin;
    bool read = digitalRead(pin);
    state = 0;
    if (digitalRead(pin)) state = _BV(DEBOUNCED_STATE) | _BV(UNSTABLE_STATE);
    previous_millis = millis();
}

void Bounce::interval(uint16_t interval_millis) {
    this->interval_millis = interval_millis;
}

bool Bounce::update() {
    // Read the state of the switch in a temporary variable.
    bool currentState = digitalRead(pin);
    state &= ~_BV(STATE_CHANGED);

    // If the reading is different from last reading, reset the debounce counter
    if ( currentState != (bool)(state & _BV(UNSTABLE_STATE)) ) {
        previous_millis = millis();
        state ^= _BV(UNSTABLE_STATE);
    } else
        if ( millis() - previous_millis >= interval_millis ) {
            // We have passed the threshold time, so the input is now stable
            // If it is different from last state, set the STATE_CHANGED flag
            if ((bool)(state & _BV(DEBOUNCED_STATE)) != currentState) {
                previous_millis = millis();
                state ^= _BV(DEBOUNCED_STATE);
                state |= _BV(STATE_CHANGED);
            }
        }

    return state & _BV(STATE_CHANGED);
}

bool Bounce::read() {
    return state & _BV(DEBOUNCED_STATE);
}
```

```
bool Bounce::rose() {
    return ( state & _BV(DEBOUNCED_STATE) ) && ( state & _BV(STATE_CHANGED));
}

bool Bounce::fell() {
    return !( state & _BV(DEBOUNCED_STATE) ) && ( state & _BV(STATE_CHANGED));
}
```

7.2 bounce.h

```
#ifndef BOUNCE_H
#define BOUNCE_H

#include <inttypes.h>

class Bounce {
 public:
    // Create an instance of the bounce library
    Bounce();

    // Attach to a pin (and also sets initial state)
    void attach(int pin);

    // Sets the debounce interval
    void interval(uint16_t interval_millis);

    // Updates the pin
    // Returns 1 if the state changed
    // Returns 0 if the state did not change
    bool update();

    // Returns the updated pin state
    bool read();

    // Returns the falling pin state
    bool fell();

    // Returns the rising pin state
    bool rose();

 protected:
    unsigned long previous_millis;
    uint16_t interval_millis;
    uint8_t state;
       uint8_t pin;
};

#endif
```

7.3 buzzer.cpp

```
#include "Arduino.h"
#include "buzzer.h"

#define OFF_STATE 0
```

```
#define ON_STATE 1
#define ON_DURATION_STATE 2

Buzzer::Buzzer()
  : previous_millis(0)
  , duration_millis(1000)
  , state(0)
  , pin(0)
{}

void Buzzer::attach(uint8_t pin) {
  this->pin = pin;
  state = OFF_STATE;
  previous_millis = millis();
}

void Buzzer::onDuration(uint16_t duration_millis) {
  on();
  state = ON_DURATION_STATE;
  this->duration_millis = duration_millis;
  previous_millis = millis();
}

void Buzzer::update() {
  if(state == ON_DURATION_STATE) {
    if (previous_millis + duration_millis <= millis()) off();
  }
}

void Buzzer::on() {
  state = ON_STATE;
  digitalWrite(pin, HIGH);
}

void Buzzer::off() {
  state = OFF_STATE;
  digitalWrite(pin, LOW);
}
```

7.4 buzzer.h

```
#ifndef BUZZER_H
#define BUZZER_H

#include <inttypes.h>

class Buzzer {
  public:
    Buzzer();
    void attach(uint8_t pin);
    void onDuration(uint16_t duration_millis);
    void on();
    void off();
    void update();

 protected:
    unsigned long previous_millis;
    uint16_t duration_millis;
    uint8_t state;
    uint8_t pin;
};

#endif
```

7.5 ledRGB.cpp

```
#include "Arduino.h"
#include "ledRGB.h"

#define OFF_STATE 0
#define ON_STATE 1
#define ON_DURATION_STATE 2

LedRGB::LedRGB()
  : previous_millis(0)
  , duration_millis(1000)
  , state(0)
  , red_pin(0)
  , green_pin(0)
  , blue_pin(0)
  , red(60)
  , green(60)
  , blue(60)
{}

void LedRGB::attach(uint8_t red_pin, uint8_t green_pin, uint8_t blue_pin) {
  this->red_pin = red_pin;
  this->green_pin = green_pin;
  this->blue_pin = blue_pin;
  state = OFF_STATE;
  previous_millis = millis();
}

void LedRGB::setColor(uint8_t red, uint8_t green, uint8_t blue) {
  this->red = red;
  this->green = green;
  this->blue = blue;
}

void LedRGB::setRedColor(uint8_t red) {
  this->red = red;
}

void LedRGB::setGreenColor(uint8_t green) {
  this->green = green;
}

void LedRGB::setBlueColor(uint8_t blue) {
  this->blue = blue;
}

void LedRGB::onDuration(uint16_t duration_millis) {
  on();
  state = ON_DURATION_STATE;
  this->duration_millis = duration_millis;
  previous_millis = millis();
}

void LedRGB::update() {
  if(state == ON_DURATION_STATE) {
    if (previous_millis + duration_millis <= millis()) off();
  }
}

void LedRGB::on() {
  state = ON_STATE;
  analogWrite(red_pin, red);
  analogWrite(green_pin, green);
  analogWrite(blue_pin, blue);
}

void LedRGB::off() {
  state = OFF_STATE;
  analogWrite(red_pin, 0);
  analogWrite(green_pin, 0);
  analogWrite(blue_pin, 0);
}
```

7.6 ledRGB.h

```
#ifndef LED_H
#define LED_H

#include <inttypes.h>

class LedRGB {
  public:
    LedRGB();
    void attach(uint8_t red_pin, uint8_t green_pin, uint8_t blue_pin);
    void setColor(uint8_t red, uint8_t green, uint8_t blue);
    void setRedColor(uint8_t red);
    void setGreenColor(uint8_t green);
    void setBlueColor(uint8_t blue);
    void onDuration(uint16_t duration_millis);
    void on();
    void off();
    void update();

 protected:
    unsigned long previous_millis;
    uint16_t duration_millis;
    uint8_t state;
    uint8_t red_pin;
    uint8_t green_pin;
    uint8_t blue_pin;
    uint8_t red;
    uint8_t green;
    uint8_t blue;
};

#endif
```

7.7 IQPCode.ino

```
#include <SoftwareSerial.h>
#include <U8glib.h>
#include <WiFlyHQ.h>
#include <ArduinoJson.h>
#include "bounce.h"
#include "ledRGB.h"
#include "buzzer.h"

#define BLUE_LED_PIN (uint8_t)9
#define RED_LED_PIN (uint8_t)10
#define GREEN_LED_PIN (uint8_t)11

#define BUZZER_PIN (uint8_t)8

#define BUTTON1_PIN A3
#define BUTTON2_PIN A1
#define BUTTON3_PIN A2
#define BUTTON4_PIN A0
//
//#define WIFI_SSID "SR"
//#define WIFI_PASSWORD "00000000"
//#define URL_HEAD "192.168.1.2"

//SoftwareSerial wifiSerial(13, 12);

//WiFly wifly;

Bounce debounce1 = Bounce();
Bounce debounce2 = Bounce();
Bounce debounce3 = Bounce();
Bounce debounce4 = Bounce();

LedRGB led = LedRGB();
Buzzer buzzer = Buzzer();

U8GLIB_NHD31OLED_2X_GR u8g(6, 5, 2, 4, 3);
//
boolean refresh = true;
//boolean messageAvailable = false;
//
char inBuf[128];
char outBuf[128];
uint8_t outBufInd = 0;
```

```
StaticJsonBuffer<200> jsonBuffer;
//char json[] = "input";

void setup() {
  pinMode(RED_LED_PIN, OUTPUT);
  pinMode(GREEN_LED_PIN, OUTPUT);
  pinMode(BLUE_LED_PIN, OUTPUT);

  pinMode(BUZZER_PIN, OUTPUT);

  pinMode(BUTTON1_PIN, INPUT_PULLUP);
  pinMode(BUTTON2_PIN, INPUT_PULLUP);
  pinMode(BUTTON3_PIN, INPUT_PULLUP);
  pinMode(BUTTON4_PIN, INPUT_PULLUP);

  debounce1.attach(BUTTON1_PIN);
  debounce2.attach(BUTTON2_PIN);
  debounce3.attach(BUTTON3_PIN);
  debounce4.attach(BUTTON4_PIN);
  led.attach(BLUE_LED_PIN, GREEN_LED_PIN, RED_LED_PIN);
  buzzer.attach(BUZZER_PIN);

//  wifiSerial.begin(9600);
//  wifly.begin(&wifiSerial, &Serial);
//
//  if(!wifly.isAssociated()) {
//    /* Setup the WiFly to connect to a wifi network */
//    wifly.join(WIFI_SSID, WIFI_PASSWORD, true);
//    wifly.save();
//  }
//
//  if (wifly.isConnected()) wifly.close();
//
//  connected = wifly.open(URL_HEAD, 80);
}

void draw() {
  u8g.setFont(u8g_font_5x7);
  u8g_prepare();
  u8g.setColorIndex(1);
  u8g.drawStr(0, 0, "Worcester Polytechnic Institute");
  u8g.drawStr(196, 0, "Wifi:");
  u8g.drawStr(226, 0, "None");
  u8g.drawHLine(0, 9, 255);
  u8g.setFont(u8g_font_6x10);
  u8g_prepare();
  u8g.setColorIndex(2);
  u8g.drawStr(0, 12, "Button1:");
  u8g.drawStr(54, 12, debounce1.read() ? "true" : "false");
  u8g.drawStr(128, 12, "Button2:");
  u8g.drawStr(182, 12, debounce2.read() ? "true" : "false");
  u8g.drawStr(0, 24, "Button3:");
  u8g.drawStr(54, 24, debounce3.read() ? "true" : "false");
  u8g.drawStr(128, 24, "Button4:");
  u8g.drawStr(182, 24, debounce4.read() ? "true" : "false");
//  u8g.drawStr(0, 36, "Fetched message:");
//  u8g.drawStr(0, 48, buf);
}

void loop() {
  refresh |= debounce1.update();
  refresh |= debounce2.update();
  refresh |= debounce3.update();
  refresh |= debounce4.update();
  led.update();
  buzzer.update();


  if(refresh) {
    u8g.firstPage();
    do {
      draw();
    } while(u8g.nextPage());
    refresh = false;
  }
}


void u8g_prepare(void) {
  u8g.setFontRefHeightExtendedText();
  u8g.setDefaultForegroundColor();
  u8g.setFontPosTop();
}
```