Team 4: The V-It

Abstract

Our device, the V-It, is a handheld reaction game similar to the “Bop It!”, but instead it replaces the auditory cues with visual and tactile cues for people that are deaf or hard of hearing. It is a fun game that also tests players’ reaction times. Our group was tasked with creating a game adaptation that could be played by those who would not normally be able to participate in a traditional game. We decided to have our product geared towards the deaf and hard of hearing community because they make up such a large proportion of the population. However, anyone that can see the LED’s can play the V-It. We designed our game to have three independent buttons that all correspond to a specific colored LED. The player has a certain amount of time to trigger the correct button after the LED flashes, and the game is over if they take too long or use the incorrect button. A vibrational motor also pulses when the LED flashes. All of our components are housed in a durable and ergonomic 3d printed casing. Through the process of prototyping, we created a functioning product. However, we used the testing and analysis process to determine that the durability and the speed of the code needed to be improved. We had a more durable casing printed and tweaked the code so the player experience was improved. We believe we were ultimately successful in accomplishing our goal. Through design cycles we were able to create a functioning and polished final product that executed the game smoothly.

Project Scope

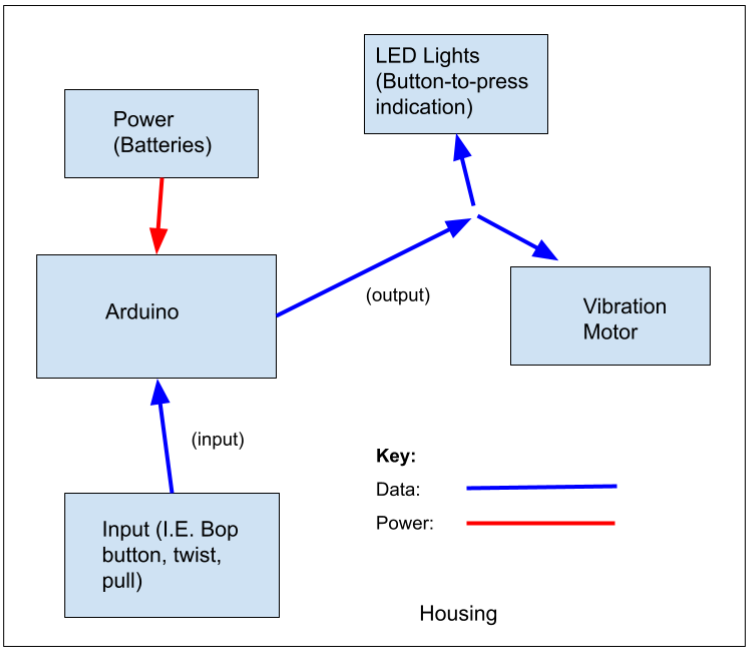
We decided to make a game that could be adapted for children with physical disabilities. In particular, children who are deaf or hard of hearing. Our issue was that because The Bop It! is an audio-based product, it fails to cater for the deaf and hard of hearing community. As a result, we decided to innovate family game night and develop a product that caters to an audience that is sometimes overlooked, which is approximately 630 million people. Thus we created the V-it, which uses lights, and vibration to make it more available to a broader audience.

Design Requirements

After interviewing our client, we discovered that, first and foremost, the electronic connections that we had in mind can wear away/break with vibration, necessitating the use of strong connections. Second, for vibration input, we could use small offset motors. Third, we must prioritize the long-term reliability of all of our materials and parts. Since we prioritized ease of use in our project, it was critical that the V-It be comfortable to use. We went through the modeling process of developing a design, creating a prototype, checking how the product feels when holding it, and then remaking it. We agreed after several attempts that the V-it should have a handle and rounded edges to make it easy to grip.

Design Alternatives and Process

Many changes have been made from our original design. The first CAD for the case looked like a Z-shape model. We 3D printed the bottom case of the first design and after discussion, decided that if we printed the top case it would be too big for the printer. The original dimension of the combined height is 3.2 inches, so we decided that half the height would be fine to fit the Arduino board, putting it at 1.6 inches. The width was also a bit too big for a child to hold, but we didn't want to change it in order to fit the arduino board. Our solution was to add a handle instead. Later, we changed our case from a Z-shape to a V-shape, because although we originally wanted four buttons to play, we decided three would be more practical. LED lights and a vibration motor work as the outputs in our V-It. After buying an original Bop-It and taking it apart we were able to study the mechanisms behind the twist-it and pull-it buttons. They need an assembly that triggers a button, so we decided to keep the housing around the buttons intact.



Final Design Choice

For the final design of our product, we chose an arduino to control our three LED's, three input buttons, and a vibrational motor, all housed in a 3D printed case.

Arduino- The V-It is run off of an arduino that is plugged into a 9V battery pack. We had originally considered creating

our own electronic board to run the V-It game, but chose an arduino because we thought it would be easier to write code directly to the board and test quickly. The arduino is then connected to each component of the game (LEDS, Input buttons, Vibrational motor) with soldered connections on the attached proto shield. The game then runs off of ~300 lines of code uploaded to the arduino.

The Programming- The programming works by first initializing which pins on the arduino board correspond to which parts of the V-It game, and other variables such as the different states of buttons or variables to keep track of the time passed. Then, at the start-up of the arduino, certain button states are set, and the position of the analog joystick is recorded, so that we can tell if it's position is moved as part of the game. Then, the arduino will begin it's runtime state and loop through different conditions that determine what is going on in the game. To start, the arduino will do nothing, and wait for the player to press a button to start the game triggering the start of a "round". At the start of each round, a random number will be generated and used to determine which of the three buttons (flick-it, twist-it, or pull-it) will need to be pressed, and the corresponding LED will flash to indicate that to the player. Each loop the arduino takes through the code, the time since the LED flashed will be recorded and used to decide different conditions of the game. Originally, this time was recorded by a separate variable, and the code had to tediously count the time passed, until we met with an ITLL engineer (Lauren Darling), who told us about the "millis" function, which can measure the amount of time passed since the start-up of the arduino. If the time passed is greater than a preset time limit allowed for each round, the player loses the round. Similarly, if the time has not run out, but the player presses a different button than the one indicated, they also lose the round. The player wins the round by pressing the correct button within the time limit. When the player wins a round, their score increases, the time limit they have to press the next button decreases, and the program gets another random number, starting a new round. When the player eventually loses, the program flashes the lights to indicate the player has lost, and then the lights flash separately, communicating the player's score. Additionally, the program returns to its idle state until one of the buttons is pushed, starting a new game.

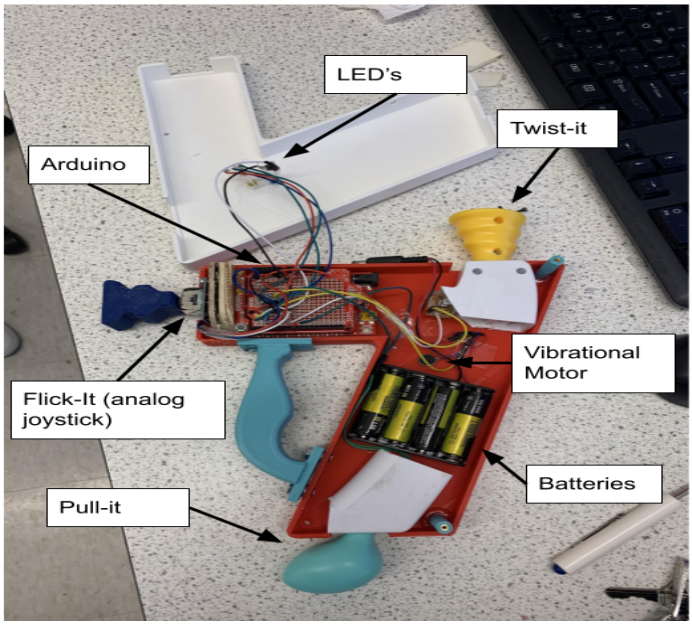
LED's- To communicate the button that the user needs to press in order to win the round, LED's light up with a color matching the button that they indicate needs to be pressed. As a team, we decided that the best position for the lights would be by the center of the device for added challenge, because having the lights near their respective button would make it easier for the player.

Pull-It, Twist-It, and Flick-It- In our original brainstorms, wanted 4 or 5 different buttons, but settled on 3: the "pull-it", "twist-it", and "flick-it". When the "pull-it" and

“twist-it” buttons are pressed, their button’s state in the arduino goes from 1 to 0, recording that they are pressed. We decided to take these buttons and their housings from a preexisting bop-it instead of building our own, because the housing was aligned in a specific way that would have been difficult to replicate otherwise. For the “flick-it” button, we decided to use an analog joystick. When the joystick’s position changes, it is recorded as another form of input, similar to the buttons.

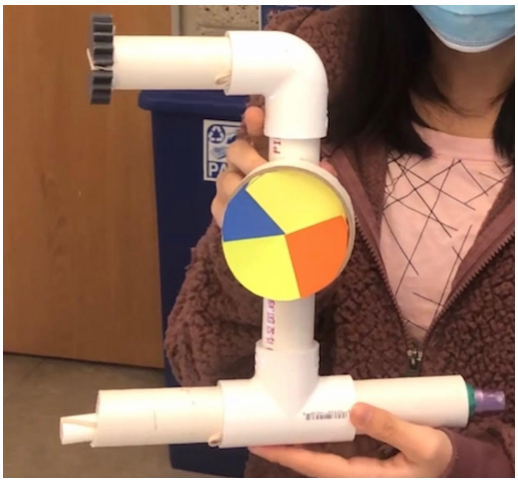
Vibrational Motor- In addition to the visual feedback created by the LED’s, we decided to add a vibrating motor to the V-It as tactile stimulation. This works at the same time as the LED’s to communicate to the player that an input is required for the game.

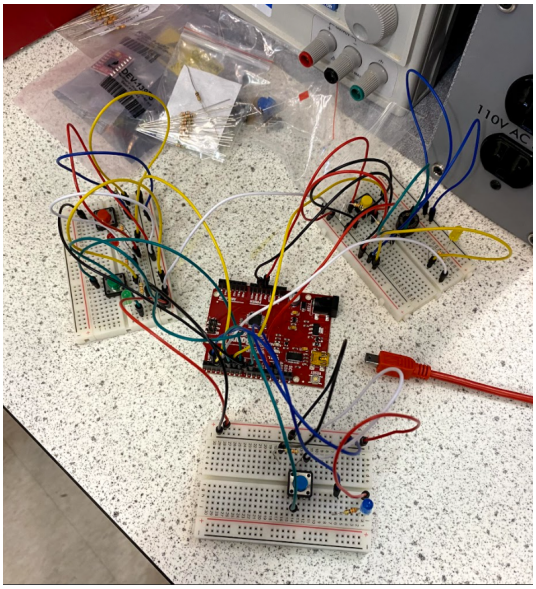
Housing- For our housing of the V-It, we decided to 3D print a case using CAD models we created. Once printed, we used a mix of hot glue, 3D printed posts, screws, and heat inserts to mount all of our components and put the two halves and handle of the design together. The screws and heat inserts allow for the case to easily be taken apart so batteries can be replaced or the design can be worked on. Originally, we had a simple design to the handle, but after it was accidentally ripped off of our prototype, we created a stronger and more intuitive design to the case.



Prototypes

In the process of our design, we had 3 prototypes. One prototype was our rapid prototype, made out of pvc pipe segments and other miscellaneous items found around the ITLL. We used this prototype as a way to hold the product and decide what we wanted in our later designs.





Another prototype we created was our circuit prototype. This prototype included three bread boards attached to our arduino, which was connected to LEDs, buttons, and a buzzer (to represent our future vibrational motor). We used this prototype to build and test the functionality of our code for the game.

Our last prototype was our first design with a 3D printed case and the final buttons we would use. We used this prototype to test the V-It case as well as ideas for how to connect the two halves. This design also unintentionally revealed that we needed to redesign the handle on our 3D printed case when it was accidentally ripped off.

<u>Critical Components</u>

Our Critical Parts for the project were our LED's, our button inputs, and our code.

LED's- Our LED's serve to communicate to the player which button they need to press, and the player would be clueless without them. As a team, we considered placing the LED's closer to their respective button, but decided that having them closer to the center would add more challenge to the game.

Buttons- Our buttons are used to get input from the players, and without them, the player would have no way of playing the game. To test these buttons beforehand, we

wrote small segments of code that tested each individual button and made sure we could get feedback from them. The flick-it button took extra time to get working, because we needed to write a function of code that took any movement from the joystick's initial position and turn it into a readable input of 1 or 0.

The Code- The code of our project was in integral part of the design of the V-It. We originally tested the code by building a circuit with simple buttons and lights to represent the final design. Once the code was completed, we tweaked different values such as the time limits the players have, and added additional functions such as lights to indicate the players score.

Final Product Development

We visualized how comfortable it will be when the user holds the toy. We decided that the flick-it should be on the left, the twist-it on the top, and the pull-it on the bottom. The user will face directly the top part of the case when they hold it from their left hand. We designed that the lights should be at the middle part of the top case to guide the user. There must be holes that are big enough for the LED lights to stick out. The height of the case is 1.6 inches so it is not too big for a kid. We stole two parts from the original Bop-it: the twist-it and pull-it, and decided to make our own flick-it. So we want to make sure that those holes will fit those parts. The case is supposed to cover the Arduino, battery, and wires perfectly. And because we want the user to be able to use this not just for once, so using screws and bolts is the best way to close and open the case when they need to change the battery. The first handle design was rounded and can be slippery when holding it and to connect it to the case we just use a hot glue gun, which is not very effective and it can fall out. So the final design for the handle we added a curve grip and holes that let it connect the case by bolts directly. The LED lights correspond to the components' colors. We use yellow, blue, and red. Yellow is for twisting, blue is for pulling, and red is for flicking. Because we 3D printed the flick-it blue, we had to spray paint it red. The V-it needs to have a logo, so we spray painted black on the top case saying 'V-It!' It was too bad that we couldn't show our last prototype in class and use it as a thumbnail. But luckily, in the presentation for the judges, we were able to show it. Our team agrees that the last frame for this project is the one that looks best out of the three tries. The first one was too big and it was a Z-shape product. The second one was blue and makes it look dark and not quite interesting, also the handle is not too well designed. For the last one, we were satisfied with the CAD design and requested the ITLL to print them out in all white, but they said the color is not guaranteed. When we got the case from the ITLL it had a top case as white color, the bottom case as an orange-reddish color, and light blue handle. We were going to paint it

all white but the longer we look at it the more interesting it seems, so we made no color changes for our final frame.

Analysis & Testing

We utilized design loops to reach our final design. We started with a simple pvc shape which caused us to think about the ergonomics of the V-It. We also used breadboards to prototype the circuitry and code used for the arduino. We purchased and disassembled a Bop It!. Analysis of how each of the buttons worked led us to use two of the button mechanisms from the bop it for the V-It. CAD modeling was used to create the first casing which fit together well, but could have been more durable and easy to hold. We used this information to print a more durable casing with screw closures that had a much better handle.

Although we didn't have any quantitative plots for our project due to the nature of it, we still used numbers to perfect the code. This was especially useful for the "flick" button because it output position values to the arduino, so we had to find a good change in position to activate that function so the arduino would detect the user input.

We used our own testing of our product as well as other input from those in the ITL to change the time intervals for the game so that it was more engaging and challenging. Initially, the game was slow and would get boring, but a shorter initial time interval and shorter time intervals as the player progressed made the V-It more challenging and fun. We planned on testing our product with Lauren Darling from the ITL because she was hard of hearing and had good input on design, but she wasn't on campus at all in April. Ultimately most of the testing was done by our own group. Through our own testing we were able to break one of our own buttons, so we fortified the mount for it so the final product was stronger.

Budget and Bill of Materials

We used many of the Arduino parts from the kit that we bought in this class, including LED lights, batteries, and wires. Additional parts that we had to buy were a vibration motor and a joystick. The vibration motor was about $1.40 and the joystick was about $1.20. We also bought a Bop-It, which is about 15 dollars. Additionally, we bought the 3D filament to print out our CAD, costing 25 dollars. We ordered our last 3D prototype from the ITLL staff to print, costing us 6.25 dollars. Lastly, we bought two spray paint colors, red and black. It costs about 6 dollars each. The overall additional cost for this project was less than 65 dollars.

Timeline

Although this was a semester long project, sometimes it felt as though there wasn't enough time to finish the project. We used a Trello board in order to set deadlines for ourselves in order to stay on track. The biggest deadline we had was to have a completed and polished product by April 23rd in order to present to judges, but there were many little deadlines before this point. We utilized presentation days as progress checkpoints in order to make sure we were on the right track fitting the course. To go into the specifics of the presentations, the preliminary design review occurred February 24th, the prototype expo occurred March 10th, the critical design review occurred 3/19, and the functional product progress occurred April 2nd. With an abundance of presentations set for us, we had motivation to further develop our product in between these time frames so that there was new material in each presentation. Aside from presentation deadlines, we also set deadlines for the group members. This included certain workshops that each member needed to complete, and when the workshop needed to be completed by. We created these goals in order to develop a variety of skill sets that we saw necessary for the product development. These assignments were divided amongst team members, but we ensured that at least two people took the same workshop. This helped as two people could work on one portion of the project instead of just one. By the end of March 19th, all the workshops we sought to take were completed. Once these were completed, we were more focused on the mini goals we set for ourselves. These ranged from little chores such as creating a name for our project all the way to 3D printing the final frame of the game. Although it was hard to follow the Trello deadlines, we all held each other accountable as the completion of one task permitted availability to work on the next. As we went through multiple loops within the design process, some of the goals we set before were modified in order to fit the up-to-date situations. In the end, we were able to stay on track due to keeping a shared Trello board and having class presentations as motivation.

The Future of our project:

Given the fact that we only had a semester to work on this project, there are limitations to our final product we would further develop if there were more time. We thought about using a screen to display the score and game mode. Unfortunately, there wasn't enough time to add this feature. In the future, we would add a screen for the score instead of communicating it via lights. Next, we would also add more buttons and lights in order to make the game more aesthetically pleasing and interesting. Finally, like the original Bop It game, we would add the same game variations that the original has. We were able to code in the basic game in which the player follows directions, but there were some game modes we wish we had time to add to our V-It. One game included

passing the game on to the next player and who then adds a specific function. As the game is passed onto the next player, they have to remember the pattern of the functions each person has added. This would make the game more collaborative, and it would also make our targeted audience receive the same experience that the normal Bop It provides. Although we did not have time for these developments, that does not mean there is no possibility of it. If we were to continue to develop the product or if another group of individuals continued our work, there would be some concepts that they need to grasp in order to be successful. For one, the group needs to be able to work together along with divide and conquer. Some portions of the project like developing the look of the product requires all team members, but some portions such as the soldering can be done by one person. As each member brings a different set of skills, it is easy to build upon one another's, but sometimes these skill sets can shine best when working on a portion of the product alone. It is important to make sure that everyone is working on something; therefore, setting mini goals and deadlines is essential as well. These can hold people accountable as well as motivate them. Within our particular product, there are some specific portions that only apply to the V-It. If this product were to be fully developed, one would need to ensure that all the connections are soldered well as this was a problem we ran into near the end. As the V-It is a mobile game, the inside pieces can get jumbled and disconnected. Although our product won the best within its section, it is not perfect. There are many components that could've been further developed. If this were to happen in the future, we believe it would only be successful through a successfully functioning group. Although there are many mechanics involved with further development of the V-It, none of these mechanics could progress if the team is dysfunctional. All in all, the V-It is a product that we are very proud of. It has its limitations due to our limited time frame to develop it, but in the end, it is a well produced product that can serve its clients well.

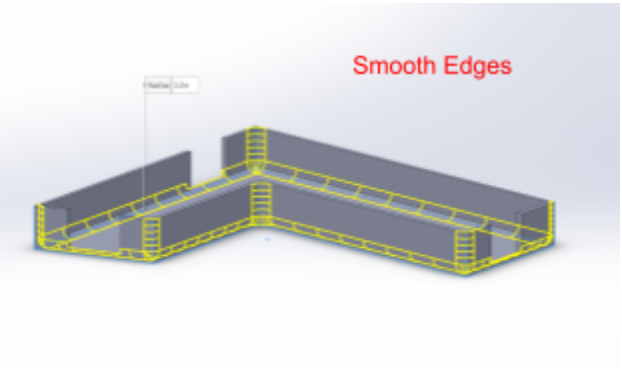
Conclusion:

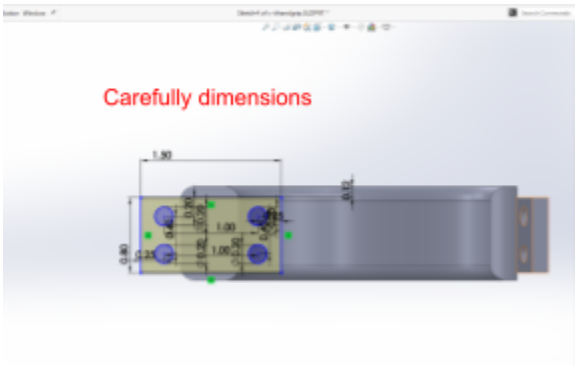
We originally set out to create a game adaptation that could be played be deaf/hard of hearing clients. Our V-It did serve this purpose by using LED lights to replace the auditory aspect. We ended up with a functioning final product that met our design criteria. Our product shows the significance of using technology to adapt and improve various products so that they can be used by audiences that would normally be excluded. Our own product could be further refined and mass produced so that it could reach a wide audience. We would have loved to add more features and game modes and would do so in the future if we continued to work on the V-it. However, we are ultimately pleased with the quality and functionality of the product that we created in the given time frame.

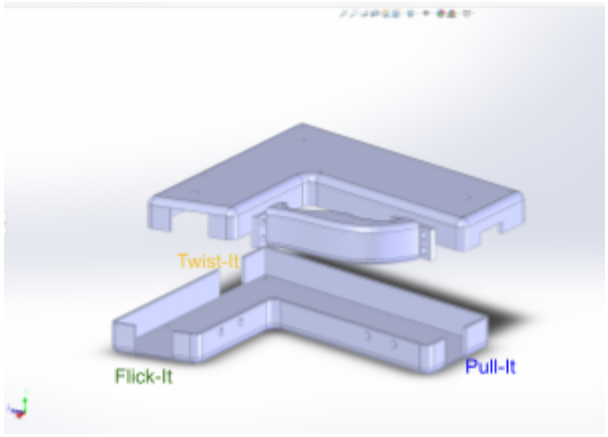
Appendix

Our Final Project:

CAD (Computer-aided Design):







Code:

BopItWithFinal

```
// Pin locations:
const int buzzerPin = 3;
const int yellowPin = 4;
const int redPin = 5;
const int bluePin = 6;
const int twistButton = 7;
const int pullButton = 8;
const int flickX = 0;
const int flickY = 5;

int curNum = 0;

// Button state variables (set to high to begin):
int pullButtonState = 1;
int twistButtonState = 1;
int flickButtonState = 1;
int curButtonState = 1; // The state of the shifting current button

// Misc Variables:
int timeNow = 0;
int timePrev = 0;
int timing = false;
int timeLim = 0;
int curTimeLim = 0;

int score = 0;
int initialX = 0;
int initialY = 0;

int hasAssigned = false;

// Function to give flick a 0 or 1 value
int flickIsPressed(int initialFlickX, int initialFlickY) { // returns 0 if "pressed", 1 if not

  if (abs(initialFlickX - analogRead(flickX)) > 100 || abs(initialFlickY - analogRead(flickY)) > 100) { // "button" pressed

    return 0;
  }
```

```
int flickIsPressed(int initialFlickX, int initialFlickY) { // returns 0 if "pressed", 1 if not

  if (abs(initialFlickX - analogRead(flickX)) > 100 || abs(initialFlickY - analogRead(flickY)) > 100) { // "button" pressed

    return 0;
  }

  else {

    return 1;
  }

} // end flickIsPressed

//function to get button state:
int getButState(int pinNum) {

  if (pinNum == 7 || pinNum == 8) { // pull or twist

    return digitalRead(pinNum);
  } // end if

  else if (pinNum == 9) { // flick

    return flickIsPressed(initialX, initialY);
  } // end else if

  else { // didn't work

    return -1;
  }

} // end getButState
```

```
//Function to flash lights when the player fails:
void failFlash(int numTimes) {

  for (int x = 0; x < numTimes; x++) {

    digitalWrite(bluePin, HIGH);
    delay(100);
    digitalWrite(bluePin, LOW);

    digitalWrite(yellowPin, HIGH);
    delay(100);
    digitalWrite(yellowPin, LOW);

    digitalWrite(redPin, HIGH);
    delay(100);
    digitalWrite(redPin, LOW);

  } // end for

} // end failFlash

// function flashes lights to indicate score to player
void scoreFlash(int curScore){

  for(int x = 0; x < curScore; x++){

      if(x%3 == 0){

        digitalWrite(bluePin, HIGH);
        delay(100);
        digitalWrite(bluePin, LOW);
      } // end if

      else if(x%3 == 1){

        digitalWrite(yellowPin, HIGH);
        delay(100);
        digitalWrite(yellowPin, LOW);
```

```
      else if(x%3 == 1){

        digitalWrite(yellowPin, HIGH);
        delay(100);
        digitalWrite(yellowPin, LOW);
      } // end if
      else{

        digitalWrite(redPin, HIGH);
        delay(100);
        digitalWrite(redPin, LOW);
      } // end if

      delay(200);

  } // end if

} // end scoreFlash

// Start of actual runtime code:

void setup() {

  Serial.begin(9600);

  timeLim = 3000; // time for first round of bop it in milliseconds, will decrease over time.
  curTimeLim = timeLim;

  pinMode(buzzerPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(redPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

  pinMode(pullButton, INPUT);
  pinMode(twistButton, INPUT);

  // setting the state to high for the 2 end buttons to work. Will be 0 when pressed
  digitalWrite(pullButton, HIGH);
  digitalWrite(twistButton, HIGH);

  digitalWrite(buzzerPin, HIGH);
  delay(500);
  digitalWrite(buzzerPin, LOW);
  delay(300);

  // set the initial X and Y positions of the joystick to base off of for the flicking
  initialX = analogRead(flickX);
  initialY = analogRead(flickY);

  // at the end of setup():

  timePrev = millis();

} // end setup
```

```
void loop() {

  timeNow = millis();

  //Serial.println(timeNow-timePrev);

  pullButtonState = digitalRead(pullButton);
  twistButtonState = digitalRead(twistButton);
  flickButtonState = flickIsPressed(initialX, initialY);

  curButtonState = getButState(curNum);

  Serial.print("X: ");
  Serial.println(analogRead(flickX));

  if (timing == false) { // the game doesn't run until a button is pressed

    //Serial.println("Not Running");

    if (pullButtonState == 0 || twistButtonState == 0 || flickButtonState == 0) {

      timing = true;
      delay(1000);
      timePrev = timeNow;
    } // end if

    curTimeLim = timeLim;

  } // the game is not running

  else { // game is running

    //Serial.print("timePrev = ");
    //Serial.print(timePrev);
    //Serial.print(" timeNow = ");
    //Serial.println(timeNow);
    //Serial.print(pullButtonState);
```

```
if ((timeNow - timePrev) >= curTimeLim) { // the round has gone past the round time limit, player lost game

  //Serial.println("Ran over time limit, lost game");

  failFlash(6);
  delay(500);
  timing = false; // end game
  curTimeLim = timeLim; // reset time limit

  hasAssigned = false;

  scoreFlash(score);
  score = 0; //reset score

} // end else if

else if (hasAssigned == false) { // the round was just reset/started, need to assign value, blink light and vibrate

  hasAssigned = true;

  //Serial.println("NEW ROUND");

  curNum = random(7, 10); // sets random pin as the button to be pressed 7 - 9

  //Serial.print("The Random num 7-9 is: ");
  //Serial.println(curNum);
  //Serial.println("(7-Twist, 8-Pull, 9-Flick)");
  // 7 - twist
  // 8 - pull
  // 9 - flick

  //Light up corresponding light and vibrate:

  if (curNum == 7) { // twist button

    digitalWrite(yellowPin, HIGH);
    digitalWrite(buzzerPin, HIGH);
    delay(500);
    digitalWrite(yellowPin, LOW);
```

```
    if (curNum == 7) { // twist button

      digitalWrite(yellowPin, HIGH);
      digitalWrite(buzzerPin, HIGH);
      delay(500);
      digitalWrite(yellowPin, LOW);
      digitalWrite(buzzerPin, LOW);
    } //end else if

    else if (curNum == 8) { // pull button

      digitalWrite(bluePin, HIGH);
      digitalWrite(buzzerPin, HIGH);
      delay(500);
      digitalWrite(bluePin, LOW);
      digitalWrite(buzzerPin, LOW);

    } // end else if

    else if (curNum == 9) { // flick button

      digitalWrite(redPin, HIGH);
      digitalWrite(buzzerPin, HIGH);
      delay(500);
      digitalWrite(redPin, LOW);
      digitalWrite(buzzerPin, LOW);

    } // end else if

    else {

      //Serial.println("Didn't light up");
    }

  } // end else if

  else if (curButtonState == 0) { // the correct button is pressed

    //Serial.println("Won Round");
```

```
    else if (curButtonState == 0) { // the correct button is pressed

      //Serial.println("Won Round");

      score = score + 1;

      if (curTimeLim > 1500) {

        curTimeLim = curTimeLim - 100;
      } // end if

      //Serial.print("Score = ");
      Serial.println(score);

      hasAssigned = false;

      delay(450);
      timePrev = timeNow;

    } // end else if

    else if (pullButtonState == 0 || twistButtonState == 0 || flickButtonState == 0) { // player presses a different button

      //Serial.println("Wrong Button Pressed");

      failFlash(6);
      delay(500);
      timing = false; // end game
      curTimeLim = timeLim; // reset time limit

      scoreFlash(score);
      score = 0; // reset score

      hasAssigned = false;

    } // end else if

    else { // no input this loop


    else if (pullButtonState == 0 || twistButtonState == 0 || flickButtonState == 0) { // player presses a different button

      //Serial.println("Wrong Button Pressed");

      failFlash(6);
      delay(500);
      timing = false; // end game
      curTimeLim = timeLim; // reset time limit

      scoreFlash(score);
      score = 0; // reset score

      hasAssigned = false;

    } // end else if

    else { // no input this loop

      //Serial.println("next");
    } // end else

  }// end else game is running

}// end loop
```