Trooper Chase

Northeastern University
Boston, MA 02115
December 8, 2017

Prepared By:

Fatema Janahi, Yael Lissack, Jillian Brislin, Matthew Colgrove

Summary

The purpose of this document is to explain the design process of the game "Trooper Chase". Trooper Chase is a game designed for a specific client. The game incorporates ideas from the franchize Star Wars to produce a fun, unique experience for the player. Stormtroopers are members of the Stormtrooper Corps, the military branch of the army of the Galactic Empire. As a member of the resistance, it is the player's duty to further the mission of justice by eliminating as many Stormtroopers as possible. The game consists of a fabric sheet with 9 faces of storm troopers on it. Behind each storm trooper mask there is a force sensor and two LED strips. When the game is started, the lights around a random stormtrooper will light up and the player must hit that stormtrooper with a lightsaber within 1.5 seconds or the stormtrooper will have the chance to escape. With each threat that the player eliminates, his/her score will go up by 10 points. At the end of the game, the player will be presented with his/her score and can choose if he/she would like to play again and try to beat the high score.

This report includes an overview of the design and creation of a game targeted towards the client, Ian Carver. To begin, the problem is introduced, followed by functions, objectives, and constraints for the solution. Next is an account of the idea generation and prototyping of the design, which were repeated several times over the course of the project. Then there is a detailed description of the final design, including information on its physical layout, circuitry, and coding. Special consideration is taken in explaining the most important aspects of the design, such as the wiring and coding of the I2C bus. An explanation of the changes made to the design over the course of constructing the final product is also included in this section. After that, the final product is analyzed to see what went well in designing and creating it and what could be improved if someone were to try to replicate it. The report ends with the conclusion that the Troopers succeeded in creating an entertaining game that they hope the client will enjoy.

Introduction to the problem

Problem Statement

Ian Carver, a second-year engineering student, was looking for a fun game that included electronic components. The game had to be played in any environment and had to be completed by the end of a six-week period. Some topics that required research were the limits and the capabilities of the Arduino Uno red boards and what additional electrical components were required for any functions that could not be done using the red board alone.

Functions	**Specifications**	**Objectives**	**Metric**	**Constraints**
Provides entertainment to client	Client can play game for at least 10 minutes without getting bored.	Should be portable.	- Can be transported by one person - Weighs less than 10 pounds	-Include electrical components included in one or more Sparkfun kits -Includes two electrical components not included in Sparkfun kits -Constructed on a budget under $40 -Includes laser-cut wooden nameplate designed on AutoCAD
Transforms electronic information code to light, sound, or other physical functions connected to the red board.	Commands stated in code match with actions performed by Sparkfun and other components.	Should be sturdy.	-Will not break/no parts will come loose while game is transported and set up	

Background Information

This game uses ideas from the fantastical world of Star Wars. The client, Ian, made clear his love of the franchise. The term Stormtrooper refers to a soldier of the Imperial army of the Galactic Empire. A Stormtrooper gives his unconditional loyalty to the Imperial regime and Emperor Palpatine. The Empire uses force, fear, and violence to control its subjects into submission. The Rebel Alliance is a resistance movement that opposes the evil ways of the Empire. As a member of the Rebellion, it is the player's duty to restore freedom to the galaxy by taking down the Empire. One way the player can do this is by eliminating the Empire's strongest weapon, its Stormtroopers. Playing Trooper Chase™ brings the galaxy one step closer to justice. To eliminate a Stormtrooper the player must hit him with a lightsaber, a weapon that uses a plasma blade to cut through virtually anything. Lightsabers can only be used by those who have had extensive training with the Force. Luckily, you, as the player, have had this training.

Method of Design

The team began with two ideas: Virtual Rock Climbing and Little Wizards. The former is a game consisting of a Twister-like mat with force sensors acting as "rocks," and the latter is a board game where the player battles a dragon and uses spells to defeat it. Diagrams of these two games can be seen in the figures below:

Figure 1 - Virtual Rock Climbing.

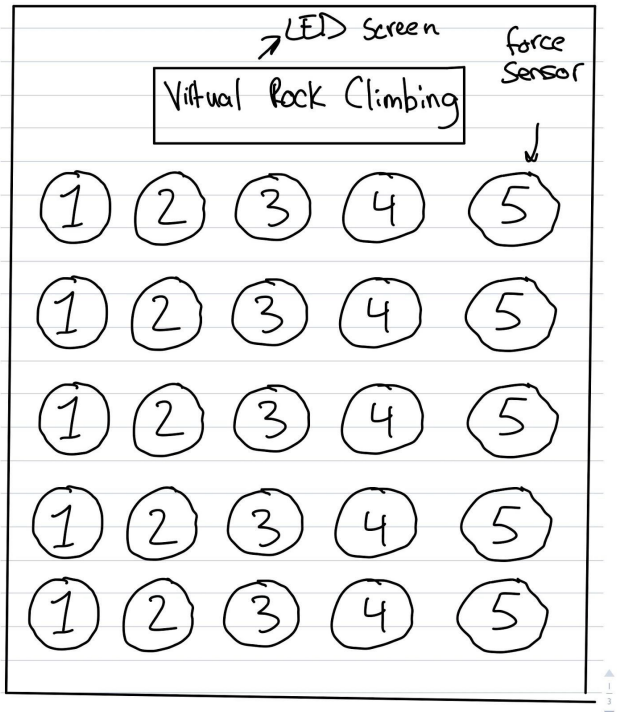
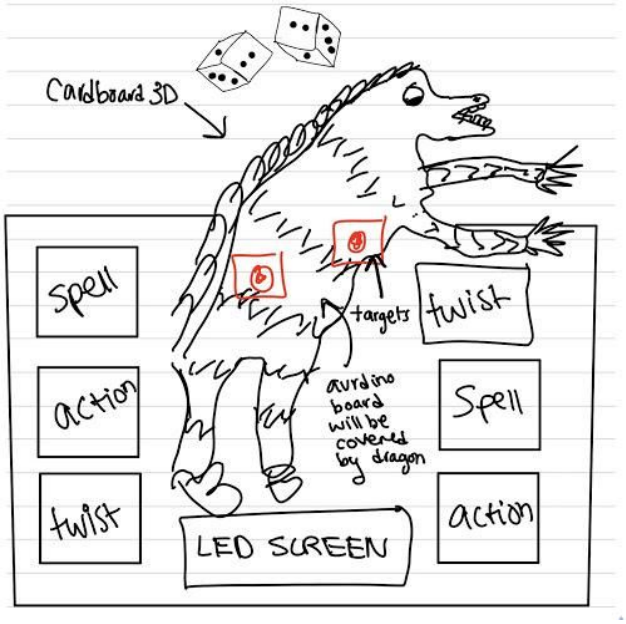
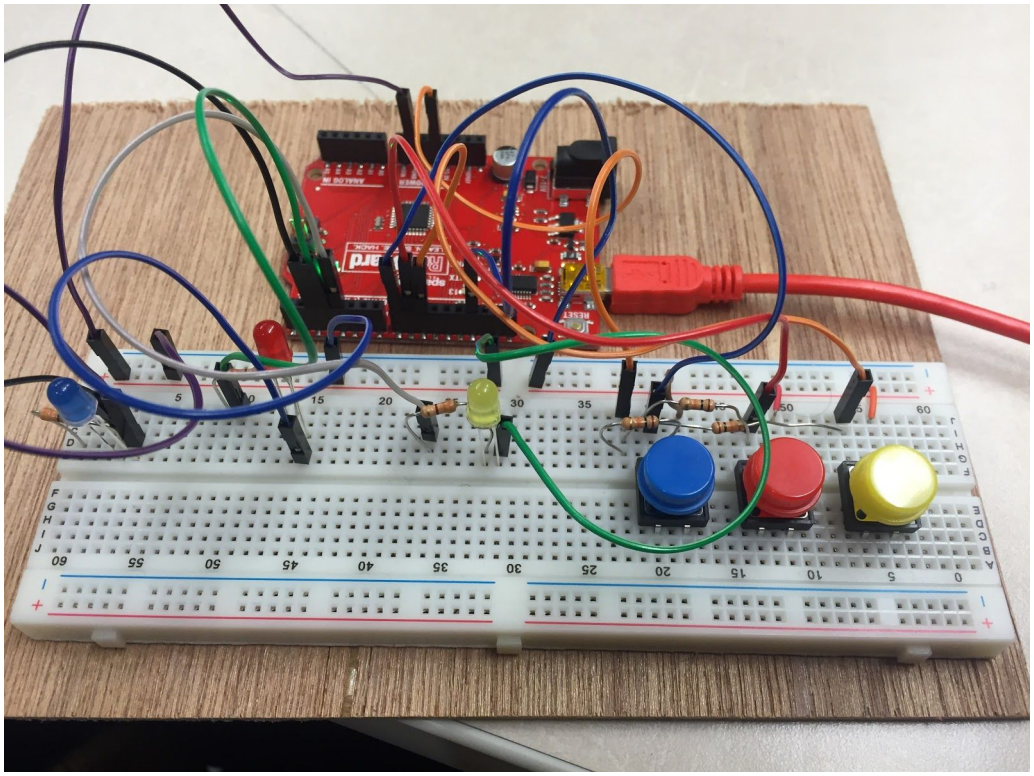


Figure 2 - Little Wizards



After debating between the two games, the team decided to accept Virtual Rock Climbing as its game. Each force sensor in this game took the place of a rock. The player "climbs" a mountain by stepping on the force sensor which prompts the system to give the user a challenge to complete. If the user successfully completes the challenge she continues on to the next level. The user wins the game by completing each challenge successfully and reaching the top of the mountain. The major issue with Virtual Rock Climbing was the administering and evaluation of the challenges. To create twenty or more interesting challenges that only require the use of force sensors and other basic technology is very difficult. Furthermore, without a way to enter the answers to riddles or questions while conforming to the restraints of the project, this game was deemed unfit. Under further examination and discussion with the client, however, the team decided to pursue a game that uses the same technology of the force sensor. This new design was comprised of twenty force sensors with LED lights strung around them on a mat similar to the one used in the game Twister. The force sensors were to act as "rocks" on a mountain. The player would position himself so that his hands and feet were on the mat. When the lights around a rock lit up, the player would have to touch the rock using his hands or feet before the lights turned off again. The player would earn points for touching the rock in time and would lose points for missing the rock or hitting the wrong rock. His/her score would be displayed on a screen on the mat. An initial prototype was created based on this design, as shown below:

Figure 3 - Initial prototype with three sets of LED lights and buttons on a breadboard.



The second prototype used three LED lights from the Sparkfun kits and three force sensors made of aluminum, cardboard, and manilla folders. The circuitry of this prototype was based off of the schematic provided by Arduino, shown in Figure 4. To create each force sensor, two pieces of aluminum foil were wrapped around circular disks of manilla folders. A piece of cardboard was cut into the shape of a ring and placed between the folders so that the two aluminum disks had no contact. When the aluminum disks were pressed such that they made contact, the circuit was completed and the light on pin 13 lit up. Once this was accomplished successfully, different colored LEDs were placed by the force sensors. A code to randomize the order of the light was written as well. To integrate the two processes, a final code was written such that when a light lit up and a force sensor was pressed, the player's score went up by 10 points. An LCD screen was also attached to the breadboard so that the player could see his/her score.

Figure 4 - Wiring schematic for the force sensors used in the second prototype.

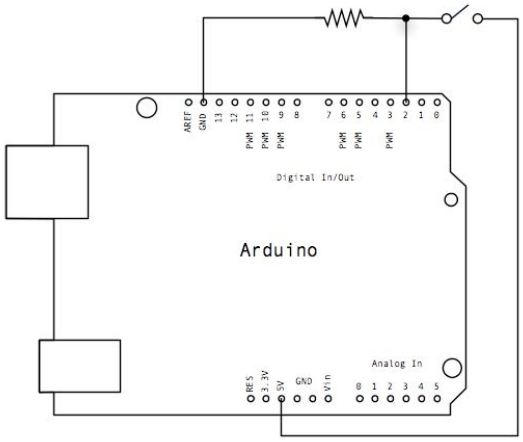
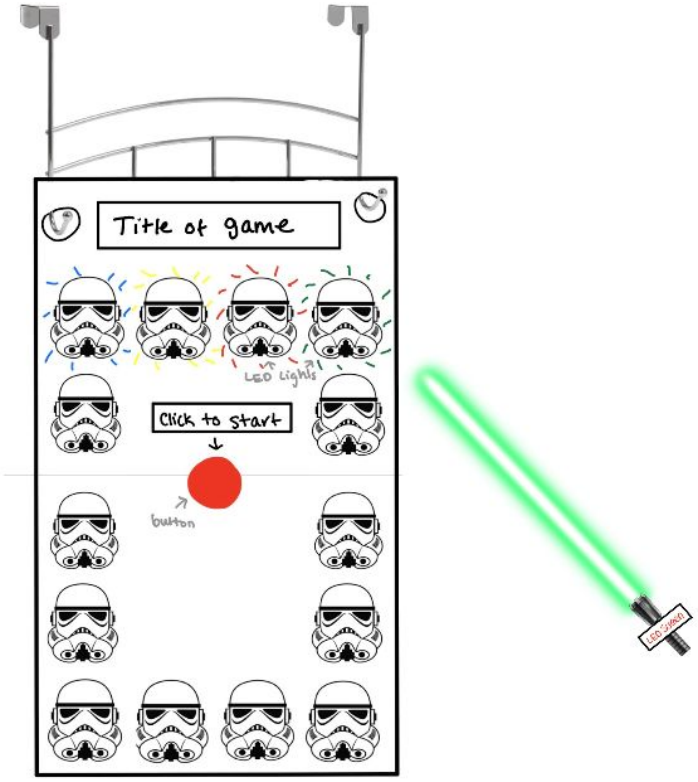


Figure 5 - Second prototype with three sets of LED lights and force sensors mounted on a cardboard platform.

After evaluating the second prototype, the team decided to slightly change the design and the theme of the game to make it more entertaining for the client. The client had previously expressed his admiration for Star Wars, so the team modified the game to incorporate the unique aspects of the Star Wars world. Instead of orienting the mat so it was laid out on the floor, the game would be a sheet of fabric hung from a door. The force sensors would be stormtroopers instead of rocks and the player would use a lightsaber to hit the stormtroopers when the lights around them lit up. An image of the first design is shown in Figure 6. The mat originally had 14 Stormtrooper masks around the perimeter, differently colored LED lights around each mask, and a large "start" button in the middle of the mat. The LCD screen was also intended to be placed on the lightsaber so the player would be able to easily see his score.

Figure 6 - Sketch of design using Star Wars theme and door hanger method.



Description of the Final Design:

The team then began working on the final product. The following is a description of the materials used, the physical aspects of the design, the circuitry, the coding, and the changes made to the design throughout the process.

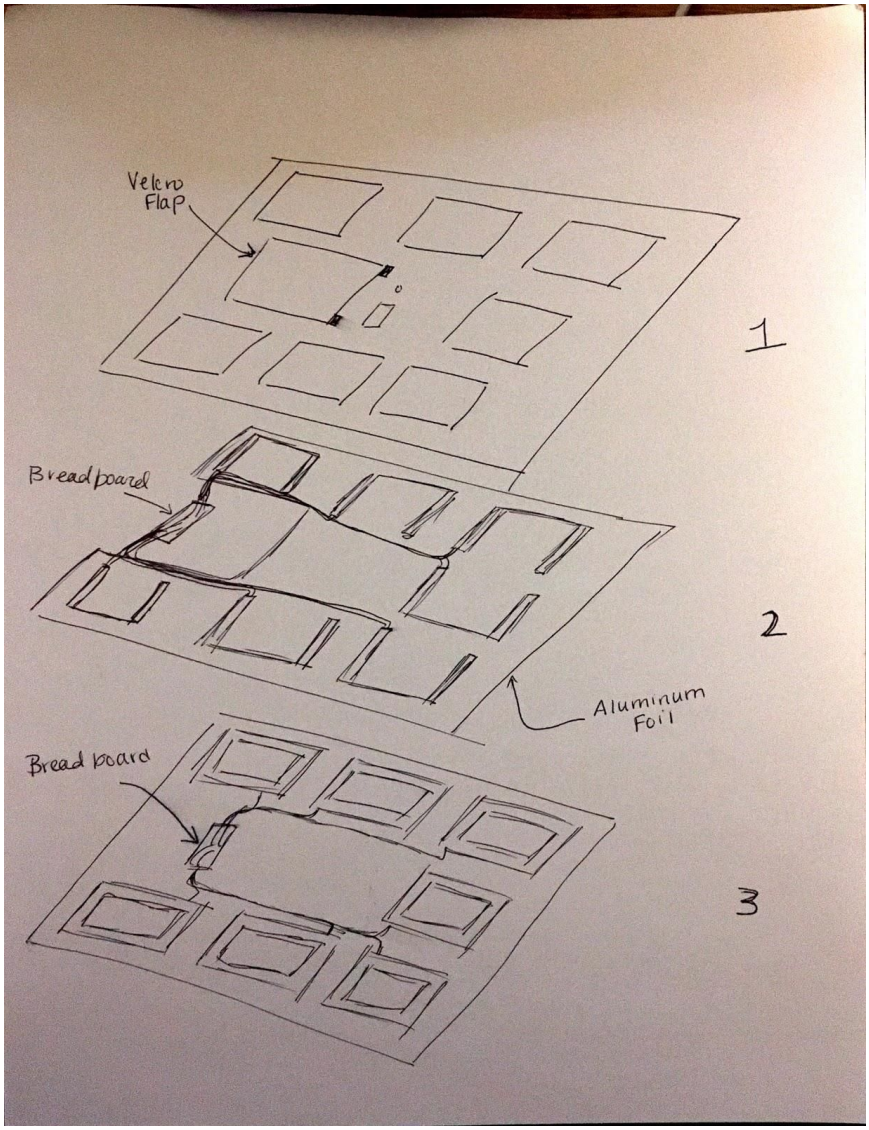
Materials

- 12V RGB LED strip (5m)
- 12V Battery
- 2 Redboards
- 3 breadboards
- Plain Fabric cut into three 40” by 30” sheets
- Star Wars Fabric (for storm trooper cutouts)
- Over-the-door hanger
- Aluminum force sensors
    - Aluminum Foil
    - Foam core
    - Electrical tape
- 9 N-channel MOSFETs
- 9V battery
- LCD Screen
- 10k ohm potentiometer
- Button

Physical Layout

The team decided to have three sheets of black fabric to compartmentalize the wiring. The first layer is what the player sees when she plays the game, with the stormtrooper masks sewn onto the black fabric. The second layer has the RGB LED lights with their wires leading to the top center of the sheet where the breadboards and red boards are. There is one strip of RGB lights above and below each mask, attached to each other via wires. Each pair of LED strips is then attached to the main breadboard. The LCD screen and start button are also on this layer, and holes are cut in the first layer of fabric so that the lights, screen, and button can be seen by the player. The aluminum force sensors are placed on the back of the second layer and the front of the third layer. The aluminum foil is glued to the fabric and the foam core is taped on top of the aluminum on the third layer of fabric. A wire is secured onto each force sensor such that the exposed wire has contact with the aluminum. Each pair of wires corresponding to one force sensor leads to the top center of the fabric were the breadboards and redboards are. The top center section of fabric on the second layer is cut out so that all of the electrical components (i.e. breadboards and redboards) are centralized. Additionally, there is a velcro flap on the top center of the first layer so that the wiring for the breadboards and red boards for the lights, screen, and force sensor can be easily accessed. A sketch displaying the layout of the layers of fabric is shown in the figure on the following page.

Figure 7 - Layout of layers with lights, screen, force sensors, and wires.



Aesthetics

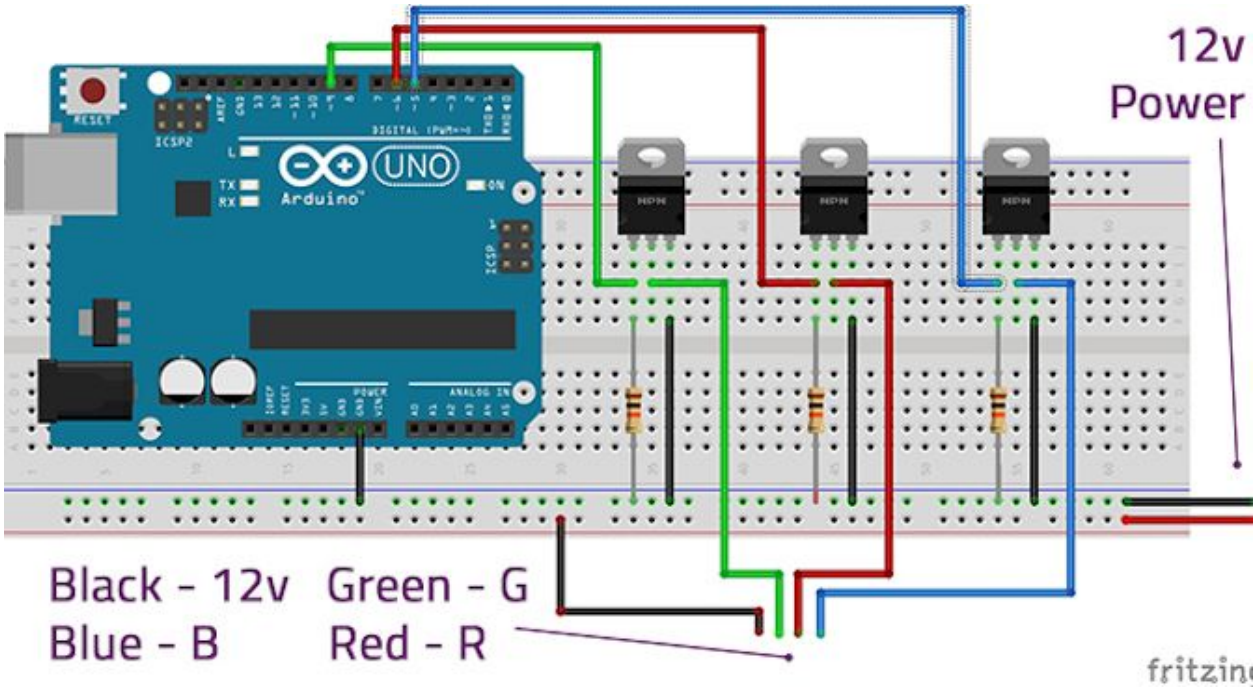
It was decided that black fabric would be used instead of a tarp as the surface on which to put the game components on because it would look nicer but would still be sturdy. The stormtroopers were cut out of a piece of Star Wars fabric and sewn onto the black fabric so that they did not look like they were just stuck on there with glue or tape. However, the overall look of the game was somewhat diminished by the holes for the lights. The holes were cut out when the fabric was laid out on a table, but when the fabric was hung on the door, gravity caused the holes on the first layer to no longer line up with the lights on the second layer. To fix this, the first and second layer had to be sewn together in various places, which gave the fabric a bunched-up look.

Circuitry

RGB LED Circuitry:

Using the external component of the RGB LED light strips instead of the LEDs in the Sparkfun kit made the circuitry slightly more complicated. Because these LED strips need 12V to operate, a 12V power source must be used. The Arduino uno, however, can only support 5V. For this reason, transistors called MOSFETs were used to prevent the redboard from receiving too much power and becoming unusable.

Figure 8 - Wiring needed for RGB LED light strips, including MOSFETs and 12V power source.



To connect the LED strips to the main redboard, the team used the diagram in Figure 8 from the website Make Use Of. Initially, the team wanted the RGB strips for different stormtroopers to change color, but because connecting all three colors for each light meant using three pins instead of one, it was decided that only one color was to be used to maximize space on the breadboard.

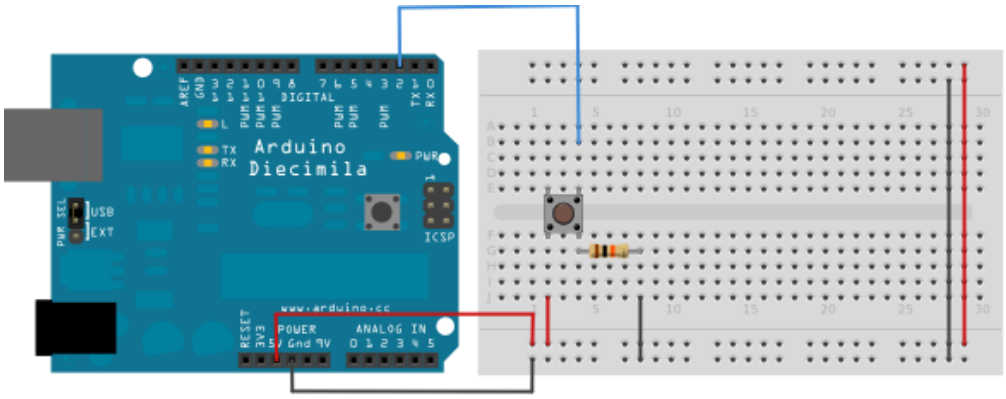
For each light, the corresponding pin was connected to the gate pin of the MOSFET, the drain pin to the LED strip (soldered to the color we selected), and the source pin to ground. This allowed the lights to use the 12V that they need, while limiting the voltage on the redboard to 5V. 10k resistors also connected the gate pin to ground as shown in Figure 8. A wire was soldered to the 12V end on each of the LED strips.

Force Sensor Circuitry

Considering the fact that buttons were used in our first prototypes, the force sensors were created such that they would replicate the switch functionality of the button. For that reason, they were connected to the arduino the same way the buttons were. A wire connected 5V to one

side of the aluminium force sensor, and the other side was wired to a resistor and back to ground, as well as to a pin.

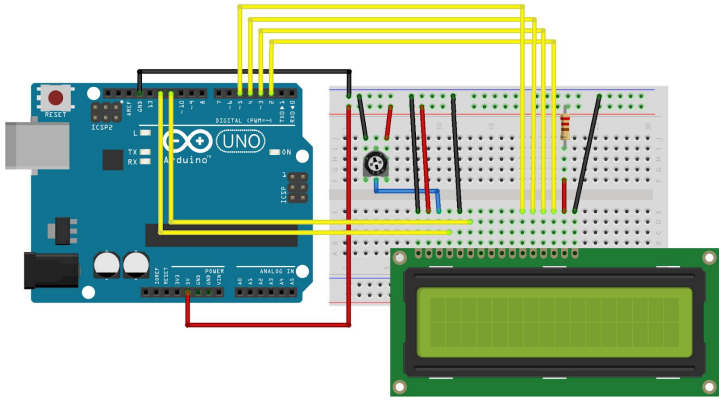
Figure 9 - Wiring required for button and the basis of the wiring for the force sensors.



LCD Screen Circuitry

The LCD screen was connected in exactly the same fashion as shown in Figure 10. For more specific information about how to do this, visit the link in citations. The setup for the LCD screen is the traditional LCD setup that can be found on the arduino website.

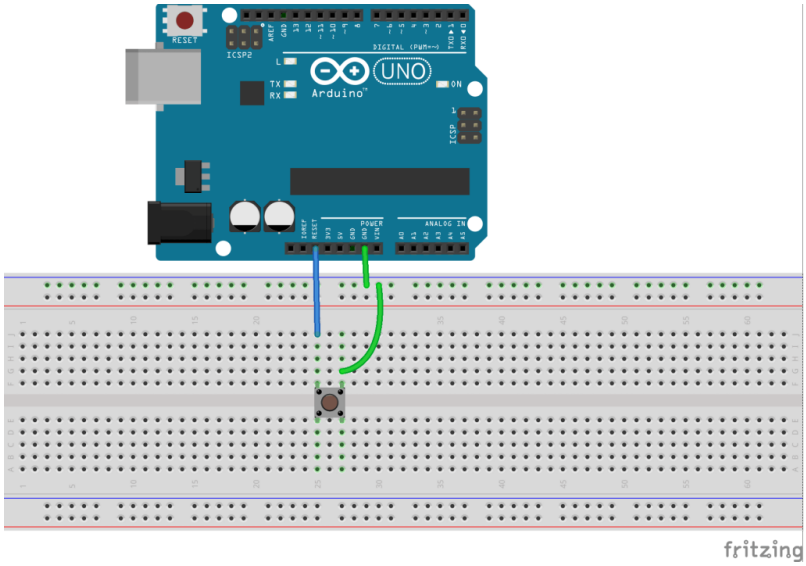
Figure 10 - Wiring required for the LCD screen.



Reset Button

The function to reset the game is essential. The setup of this function is shown in the diagram in Figure 11 and two of the red boards were connected to the same button so that all of the functions were reset. Even though this setup was quite simple, the reset button had to be accessible from the front layer of the fabric. When this button is pressed, it resets both of the redboards at the same time, ensuring that they run simultaneously.

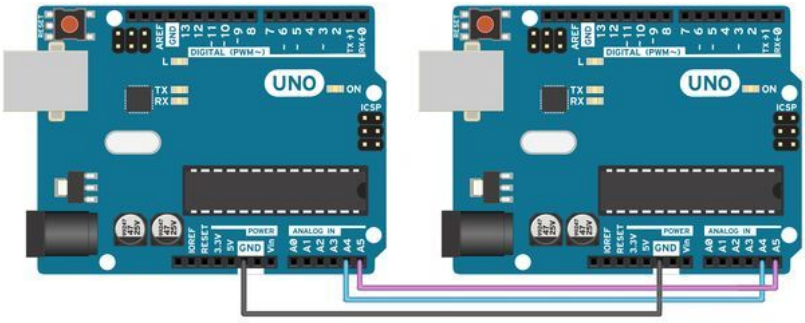
Figure 11 - Wiring required for the Reset/Start button.



I2C circuitry

The wiring for the I2C between two arduinos was relatively simple. As shown in figure 12 below, the grounds on the two red boards were connected. The figure also shows that the A4 and A5 pins were connected, but since the red boards used are slightly different than those in the figure, SDL and SDA pins were connected. It does not make any difference whether the A4 and A5 pins are connected or the SDL and SDA pins are connected. The SDL and SDA pins were chosen because that freed up the analog pins to be used for the force sensors.

Figure 12 - Wiring setup for I2C Bus.



Once the wiring was completed, it was determined which red board would be the master and which would be the slave. Any components that were connected to pins on the master red board were declared on the master code and any components that were connected to pins on the slave red board were declared on the slave code. Additional coding was needed to set up the communication between the two red boards, which is explained in the coding section. The lights

were connected to pins on the master red board and were therefore coded on the master code and the force sensors and LCD screen were connected to pins on the slave red board and coded on the slave code. There were not enough digital pins on the slave red board for the LCD screen and the nine force sensors, so four force sensors were connected to analog pins and declared as digital pins in the slave code.

Code

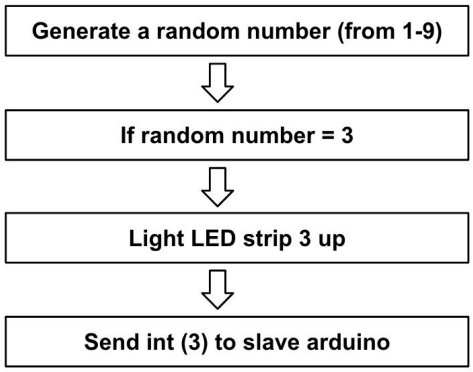
The core of the coding for this project is the use of the Wire Library to implement I2C communication between two redboards. This library allows the use of different functions to send and receive data, such as characters or integers, from one arduino to the other.

```
#include<Wire.h>
```

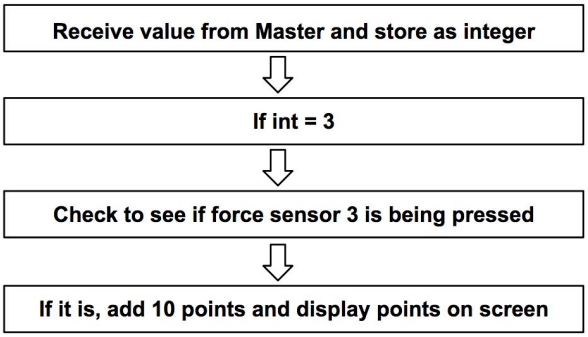
Because I2C communication was used, there were two sets of code: a master code that was uploaded onto the master arduino, and a slave code that was uploaded onto the slave arduino.

The master code directly controlled the lights and piezo speaker, whereas the slave code controlled the force sensors and the LCD screen.

Master Code (Simplified)



Slave Code (Simplified)



Master Code

To make the code truly random, the randomSeed() function was used and seeded values from analog pin 0, which was not connected to anything, in the void setup. This way, we were able to ensure that the client can play the game multiple times and have different patterns of lights so it does not get redundant.

```
randomSeed(analogRead(0));
```

The master code then begins by generating a random number between (1-9) using the random() function.

```
int randomNumber
randomNumber = random(1,10);
```

After the number is generated, the program sends the value to the Slave arduino using the Wire.write function from the Wire Library. To do so, we had to begin the transmission and input the address of the slave that we are transmitting to (5), and then end the transmission.

```
Wire.beginTransmission(5);
Wire.write(randomNumber);
Wire.endTransmission();
```

After that, through a series of if statements, the program checks to see what the random number that has been generated is, and turns on the corresponding light.

```
if (randomNumber == 1)
{
digitalWrite (LED1, 255);
digitalWrite (LED2, 0);
digitalWrite (LED3, 0);
digitalWrite (LED4, 0);
digitalWrite (LED5, 0);
digitalWrite (LED6, 0);
digitalWrite (LED7, 0);
digitalWrite (LED8, 0);
digitalWrite (LED9, 0);
}
```

After much trial and error, it was determined that because the LEDs are RGB LEDs, the values of the other lights had to be 0 to make sure they were off, otherwise they would be dimly on.

Following that, there is a 900 ms delay where the light would stay on, allowing time for the user to see and hit the storm trooper, then all the leds were turned off and had a 250 ms delay to give the user time to get back into position before the next LED lights up.

Finally, a piezo speaker was included to audibly indicate when the game time was over. This was done using the millis() function. In short, the code described above would run under an if statement that the time was under or equal to a minute, and once it exceeds that time, the speaker sounds for 2000 ms, and the program stops running.

```
unsigned long minutes = 60000;
void loop {
if (millis() <= minutes) {
// run master code
}
if ((millis() > minutes) and ( millis() <= 62000)) {
    tone(piezoPin, 100, 500); /* if one minute is over then buzz
the speaker for 2000 ms */
 }
 else
 {
 // at any other time - do nothing
 }
}
```

Slave Code

In order to receive data from the Master code, function that the Wire.onReceive() function can call on had to be created. The function, receiveEvent, read the values that were being sent from the Master redboard and stored them as an integer numberfrommaster.

```
int numberfrommaster;
void receiveEvent(int howManybytes){
  while(Wire.available())
  {
    numberfrommaster = Wire.read();
  }
}
void setup() {
Wire.begin(5);
Wire.onReceive(receiveEvent);
}
```

As can be seen from the code above, the Wire.begin() function uses the same address as the Master Code does, which is 5. This is how the redboards know to communicate with each other. Similar to the master code, the slave code also runs a series of if statements. It checks what the value of the 'numberfrommaster' is, and based on that it checks whether the corresponding force sensor is being pressed. If it is, then 10 points are added to the user's score.

```
if (numberfrommaster==1)
     {
     delay(300);
     buttonState1 = digitalRead(buttonPin1);

     if (buttonState1 == HIGH)
          {
          points = points + 10;
          }
     }
```

After going through all the if statements, the lcd screen prints the number of points to the screen. This way, the score is updated as the game is being played.

```
lcd.setCursor(0, 0);
lcd.print("Points:  ");
lcd.setCursor(0, 1);
lcd.print(points);
```

Similar to the Master code, this entire code is inside an if statement that runs only when the time is less than or equal to 1 minute. If one minute is over, however, it displays the Total Points received on the LCD screen.

```
if (millis() <= minutes) {
//slave code
}
else
{
Serial.print("Total points:  ");
Serial.println(points);
lcd.setCursor(0, 0);
lcd.print("Total points:  ");
lcd.setCursor(0, 1);
lcd.print(points);
```

```
}
```

Modifications:

In additions to the changes made to the design between the first and second prototypes, there were also many changes made to the final design during its construction. To begin, the number of stormtroopers, and therefore force sensors and RGB LED strips, was reduced from fourteen to nine because there was not enough room on the fabric to fit twenty stormtroopers and for simplification purposes. The force sensor under Darth Vader was also removed because the space was needed to store the red boards and breadboards. Also, the LCD screen was changed from being on the lightsaber to being on the fabric because it was determined that the player would not be standing that far away from the fabric and would be able to see the score on the screen if it were on the fabric. Additionally, to prevent the fabric from sagging, two strips of wood had to be glued onto the top corners of the fabric. Finally, the three layers did not line up perfectly so that the lights could not be seen through the holes cut out in the first layer. Because of this, the first fabric had to be stitched onto the second layer so that it did not sag and the lights were visible.

Analysis

When evaluating the final design and the work put into it, the team acknowledged the great quality of work that went into getting the RGB LED lights, force sensors, and LCD screen to work well individually. The team was able to wire the lights using the 12V battery and MOSFETs and code them to light up randomly very smoothly. Additionally, the force sensors worked really well on the fabric, which previously had raised concern because the only literature the team found regarding force sensors had been done using cardboard.

The largest challenge the team faced was programming all of the systems (the force sensors, LEDs, LCD screen) to work together. This was partly due to the difficulty of connecting two red boards through the I2C process and partly due to the delay functions in the coding. The code would only read the state of the force sensors (whether they were being pressed or not) at a certain time after a delay, so the team had to figure out what the delay should be so that the code checked the force sensors at the time when the player was pressing them, or was supposed to be pressing them. This required a lot of trial and error and as a result took a little longer than expected. Eventually, the delays were made very short so that the program would loop and check whether the force sensor was being pressed more than once during the time the light was on.

Furthermore, the LCD screen would sometimes haphazardly jump from increasing the score by ten if the correct stormtrooper was hit to increasing it by one hundred and then would jump back down to increasing it by ten again. This problem was particularly perplexing because the serial monitor would always display the correct score even if the LCD screen did not. Finally, the team determined this was a problem with the screen and that a better way to display the score should be investigated, such as making a “score board” using the LED strips.

Given more resources, Star Wars music and/or sound effects can be added to enhance the theme of the project and to serve as an indicator of the one-minute time limit: when the music stops playing, the player’s time is up. It would also be more enjoyable for the player if there were different sound effects for when he hits the stormtrooper and when he misses. However, due to time constraints, this was not achievable for this project.

The layout of the game could also be improved. As stated before, the Darth Vader force sensor had to be removed because there was not enough room to store the breadboards and redboards. If the Darth Vader sensor was further down on the mat, it could have been used. Moreover, as previously stated, the first layer had to be folded to line up with the second layer because the holes for the lights were cut out when the fabric was laid on a table. As a result, gravity dragged the fabric down slightly when it was hung on a door and the holes did not match up with the lights. With better planning, this could have been avoided.

Also, one of the main concerns with the current design is the fact that the player may accidentally hit the arduino red boards and breadboards if they miss a stormtrooper. This could lead to wires disconnecting or the actual breaking of breadboards. As for the game's power source, the 9v battery used to power the slave redboard was placed between the first and second layers of fabric. During testing, this did not seem like a problem as it was accessed through the Darth Vader cutout in the middle of the game. This setup, however, makes folding up the mat slightly inconvenient for the player.

Additionally, adding difficulty levels would increase the complexity of the game, and make it more fun. Overall, the team is confident that they succeeded in creating a fun, unique game for the client that adhered to the objectives and constraints of the project.

Conclusion

At this point, the team does not have any plans for continuing to work on this project. If anyone were to pick up where the team left off, it is recommend that they add music and sound effects to the game using a more advanced speaker than the one in our sparkfun kits.They should also utilize uniform pieces of fabric so all of the holes in the fabric line up well. Using a sewing machine would also help the project have a more uniform and clean aesthetic. Lastly, they should find a way to protect the arduino components from any accidental hits by the user, and make the battery accessible from the back so that it can be easily changed.

Looking back on the design process, the game changed a lot since its conception. The final product is a lot different from what the team initially envisioned. The team faced many challenges along the way, and there is definitely room for improvement. Overall, however, the team accomplished their goal of providing an entertaining game for Ian and will be open to any feedback he has.

Citations

- Pilgrim, P. (2008, November). Make your own load cell (force sensor) from stuff you already have. Retrieved December 08, 2017, from http://forums.parallax.com/discussion/108049/make-your-own-load-cell-force-sensor-from-stuff-you-already-have
- "Arduino - Button." *Arduino - Home*, www.arduino.cc/en/Tutorial/Button.
- "Arduino - HelloWorld." *Arduino - Home*, www.arduino.cc/en/Tutorial/HelloWorld.
- Buckley, Ian. "Ultimate Guide to Connecting LED Light Strips to Arduino." *MakeUseOf*, 25 May 2017, www.makeuseof.com/tag/connect-led-light-strips-arduino/.
- Finio, Ben. "Arduino-controlled RGB LED Infinity Mirror." *Instructables.com*, 21 Oct. 2017, www.instructables.com/id/Arduino-controlled-RGB-LED-Infinity-Mirror/Th.

Appendix

Time and Expense Sheet:

Name	**Time (hours)**	**Expense**
Fatema	59.5	12V Battery - $4.00 Aluminum foil-$2.00
Jilian	63.5	Fabric - $18.07 RGB LED lights - $6.50
Yael	53.5	Foam core - $4.10 Laser cut - $5.03
Matthew	49.5	
Total	222	$32.72

MOSFETS: borrowed from Fatema's friend
Over the door hanger: Fatema's
Light Saber: borrowed from Yael's friend
Speaker: borrowed from Engineering Lab

Raw Master Code

```
/*  Filename: Project_Code_MASTER
 *  Authors: Fatema Janahi, Yael Lissack, Jillian Brislin, Matthew Colgrove
 *  Brief: Code for Master Arduino
 *  Randomly lights up lights using a random number generator and
 * sends corresponding integer to Slave. When time is up, buzzer sounds.
 */

//Including Libraries
#include<Wire.h> //including the wire Library

//int to hold random number
int randomNumber;

//LED pin values
int LED1 = 5;
int LED2 = 2;
int LED3 = 3;
int LED4 = 4;
int LED5 = 6;
int LED6 = 7;
int LED7 = 8;
int LED8 = 9;
int LED9 = 10;

//piezo pin values
int piezoPin = 12;

//value of minutes before buzzer sounds
unsigned long minutes = 60000;

void setup() {

  //initiating wire library to join I2C bus
  Wire.begin();

  //setting up leds as outputs
  pinMode (LED1, OUTPUT);
  pinMode (LED2, OUTPUT);
  pinMode (LED3, OUTPUT);
  pinMode (LED4, OUTPUT);
  pinMode (LED5, OUTPUT);
  pinMode (LED6, OUTPUT);
  pinMode (LED7, OUTPUT);
  pinMode (LED8, OUTPUT);
  pinMode (LED9, OUTPUT);

  //seeding the random number generator
  randomSeed(analogRead(0));
}

void loop() {
```

```
//run this code for 1 min only
if (millis() <= minutes) {

  randomNumber = random(1,10); //generate a random number

  Wire.beginTransmission(5); //begin transmission with Slave Arduino
  Wire.write(randomNumber); //send the random number
  Wire.endTransmission();

  //Light up the lights according to the number generated

  if (randomNumber == 1)
  {
    //turn the first light on and all others off
    digitalWrite (LED1, 255);
    digitalWrite (LED2, 0);
    digitalWrite (LED3, 0);
    digitalWrite (LED4, 0);
    digitalWrite (LED5, 0);
    digitalWrite (LED6, 0);
    digitalWrite (LED7, 0);
    digitalWrite (LED8, 0);
    digitalWrite (LED9, 0);
  }

  if (randomNumber == 2)
  {
    //turn the second light on and all others off
    digitalWrite (LED1, 0);
    digitalWrite (LED2, 255);
    digitalWrite (LED3, 0);
    digitalWrite (LED4, 0);
    digitalWrite (LED5, 0);
    digitalWrite (LED6, 0);
    digitalWrite (LED7, 0);
    digitalWrite (LED8, 0);
    digitalWrite (LED9, 0);
  }

  if (randomNumber == 3)
  {
    //turn the third light on and all others off
    digitalWrite (LED1, 0);
    digitalWrite (LED2, 0);
    digitalWrite (LED3, 255);
    digitalWrite (LED4, 0);
    digitalWrite (LED5, 0);
    digitalWrite (LED6, 0);
    digitalWrite (LED7, 0);
    digitalWrite (LED8, 0);
    digitalWrite (LED9, 0);
  }

  if (randomNumber == 4)
```

```
{
  //turn the fourth light on and all others off
  digitalWrite (LED1, 0);
  digitalWrite (LED2, 0);
  digitalWrite (LED3, 0);
  digitalWrite (LED4, 255);
  digitalWrite (LED5, 0);
  digitalWrite (LED6, 0);
  digitalWrite (LED7, 0);
  digitalWrite (LED8, 0);
  digitalWrite (LED9, 0);
}

if (randomNumber == 5)
{
  //turn the fifth light on and all others off
  digitalWrite (LED1, 0);
  digitalWrite (LED2, 0);
  digitalWrite (LED3, 0);
  digitalWrite (LED4, 0);
  digitalWrite (LED5, 255);
  digitalWrite (LED6, 0);
  digitalWrite (LED7, 0);
  digitalWrite (LED8, 0);
  digitalWrite (LED9, 0);
}

if (randomNumber == 6)
{
  //turn the sixth light on and all others off
  digitalWrite (LED1, 0);
  digitalWrite (LED2, 0);
  digitalWrite (LED3, 0);
  digitalWrite (LED4, 0);
  digitalWrite (LED5, 0);
  digitalWrite (LED6, 255);
  digitalWrite (LED7, 0);
  digitalWrite (LED8, 0);
  digitalWrite (LED9, 0);
}

if (randomNumber == 7)
{
  //turn the seventh light on and all others off
  digitalWrite (LED1, 0);
  digitalWrite (LED2, 0);
  digitalWrite (LED3, 0);
  digitalWrite (LED4, 0);
  digitalWrite (LED5, 0);
  digitalWrite (LED6, 0);
  digitalWrite (LED7, 255);
  digitalWrite (LED8, 0);
  digitalWrite (LED9, 0);
}
```

```
  if (randomNumber == 8)
  {
    //turn the eightth light on and all others off
    digitalWrite (LED1, 0);
    digitalWrite (LED2, 0);
    digitalWrite (LED3, 0);
    digitalWrite (LED4, 0);
    digitalWrite (LED5, 0);
    digitalWrite (LED6, 0);
    digitalWrite (LED7, 0);
    digitalWrite (LED8, 255);
    digitalWrite (LED9, 0);
  }

  if (randomNumber == 9)
  {
    //turn the ninth light on and all others off
    digitalWrite (LED1, 0);
    digitalWrite (LED2, 0);
    digitalWrite (LED3, 0);
    digitalWrite (LED4, 0);
    digitalWrite (LED5, 0);
    digitalWrite (LED6, 0);
    digitalWrite (LED7, 0);
    digitalWrite (LED8, 0);
    digitalWrite (LED9, 255);
  }

  delay(900); // keep light on for this amount of time

  //now turn all lights off to give user time to reposition him/herself
  digitalWrite (LED1, 0);
  digitalWrite (LED2, 0);
  digitalWrite (LED3, 0);
  digitalWrite (LED4, 0);
  digitalWrite (LED5, 0);
  digitalWrite (LED6, 0);
  digitalWrite (LED7, 0);
  digitalWrite (LED8, 0);
  digitalWrite (LED9, 0);

  delay(250); // keep them off for a short time
  }

 // if the time is over (1 min), sound the buzzer for 2000 ms
 if ((millis() > minutes) and ( millis() <= 62000)) {
     tone(piezoPin, 100, 500); // if its at one minute then make the sound
 }
 else
 {
    // at any other time - no sound
 }
}
```

Raw Slave Code

```
/*
 *  Filename: Project_Code_SLAVE
 *  Authors: Fatema Janahi, Yael Lissack, Jillian Brislin, Matthew Colgrove
 *  Brief: Code for Slave Arduino
 *  Uploaded to board connected to  force sensors and LCD screen. Receives
random
 *  number from master and checks to see if corresponding force sensor is being
 *  pressed. Adds points accordingly.
 */

//Including libraries
#include<Wire.h>
#include<LiquidCrystal.h>

//setting up LCD screen
const int rs = 12, en = 11, d4 =5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

//declaring variables to hold data
int numberfrommaster;
int points = 0;

//time to run for
unsigned long minutes = 60000;

//button pin values
const int buttonPin1 = 6;
const int buttonPin2 = 7;
const int buttonPin3 = 8;
const int buttonPin4 = 9;
const int buttonPin5 = 10;
const int buttonPin6 = A0;
const int buttonPin7 = A1;
const int buttonPin8 = A2;
const int buttonPin9 = A3;

//button states set to OFF
int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
int buttonState4 = 0;
int buttonState5 = 0;
int buttonState6 = 0;
int buttonState7 = 0;
int buttonState8 = 0;
int buttonState9 = 0;

//function receives data from Master Arduino
void receiveEvent(int howManybytes)
{
    while(Wire.available())
    {
```

```
        numberfrommaster = Wire.read();
    }
}

void setup() {
    //begin serial data transmission
    Serial.begin(9600);
    //begin I2C Communication
    Wire.begin(5);
    Wire.onReceive(receiveEvent);
    //initialize screen
    lcd.begin(16,2);

}

void loop() {

    //if time is not up
    if (millis() <= minutes) {

      //depending on the number received from master, read whether that
      //button is being pressed, and increment score accordingly

      if (numberfrommaster==1)
      {
          delay(750); //time delay for user to press button

          buttonState1 = digitalRead(buttonPin1); //read button

          if (buttonState1 == HIGH) //increment points if pressed
          {
              points = points + 10;
          }
            delay(250);
      }


      if (numberfrommaster==2)
      {
          delay(750);

          buttonState2 = digitalRead(buttonPin2);
          if (buttonState2 == HIGH)
          {
              points = points + 10;
          }

            delay(250);
      }


      if (numberfrommaster==3)
      {
          delay(750);
```

```
    buttonState3 = digitalRead(buttonPin3);
    if (buttonState3 == HIGH)
    {
        points = points + 10;
    }

    delay(250);
}



if (numberfrommaster==4)
{
    delay(750);

    buttonState4 = digitalRead(buttonPin4);
    if (buttonState4 == HIGH)
    {
        points = points + 10;

    }
    delay(250);
}

if (numberfrommaster==5)
{
    delay(750);

    buttonState5 = digitalRead(buttonPin5);
    if (buttonState5 == HIGH)
    {
        points = points + 10;
    }
    delay(250);
}

if (numberfrommaster==6)
{
    delay(750);

    buttonState6 = digitalRead(buttonPin6);
    if (buttonState6 == HIGH)
    {
       points = points + 10;
    }

    delay(250);
}


if (numberfrommaster==7)
{
    delay(750);
```

```
          buttonState7 = digitalRead(buttonPin7);
          if (buttonState7 == HIGH)
          {
            points = points + 10;

          }
          delay(250);
      }


      if (numberfrommaster==8)
      {
          delay(750);

          buttonState8 = digitalRead(buttonPin8);
          if (buttonState8 == HIGH)
          {
            points = points + 10;
          }
          delay(250);
      }

      if (numberfrommaster==9)
      {
        delay(750);

        buttonState9 = digitalRead(buttonPin9);
        if (buttonState9 == HIGH)
        {
          points = points + 10;
        }
        delay(250);
      }

    //output points to LCD screen as game is played
    lcd.setCursor(0, 0);
    lcd.print("Points:  ");
    lcd.setCursor(0, 1);
    lcd.print(points);
    }
else //time is up
    {
    //display total points
    lcd.setCursor(0, 0);
    lcd.print("Total points:  ");
    lcd.setCursor(0, 1);
    lcd.print(points);
      }
}
```