
# Prototyping Interactive Systems (Midterm Project Report)

---

Malvika Bansal, Leslie Filko, Olesia Koval, Shivin Saxena

*Instructor - Dr. Stephen Voida*

# Table of Contents

# Introduction

***“One Die to rule them all, One Die to replace them,***
***One Die to play them all and in the darkness see them”***

Dice are small throwable objects with multiple resting positions, used for generating random numbers. Dice are used for a number of games ranging from gambling games to countless tabletop games. While a traditional and most commonly used die is a rounded cube, with each of its six faces showing a different number ranging from 1 to 6, specialised dice exist with irregular or polyhedral shapes, with faces marked with symbols rather than numbers, capable of producing results other than six.

A common practice seen amongst most tabletop games such as *Dungeons and Dragons*, *Risk* etc. is that players must deal with a number of different dice. It is very common for tabletop gamers to carry around their own personal set of dice and in several cases, misplace or forget them. We want to address this hassle of dealing with a collection of specialised dice.

Introducing the “OLSM die”, the one universal die capable of replacing every other die ever used in a game. The OLSM die is a product concept that enables players to choose and roll any given die using just the one electronic die. Equipped with a LCD screen, button(s), a motion sensor and a speaker, the OLSM die enables the selection of a range of die (d4, d8, d10, d20 etc.) and simulated rolling of a die after the device is given a good shake.

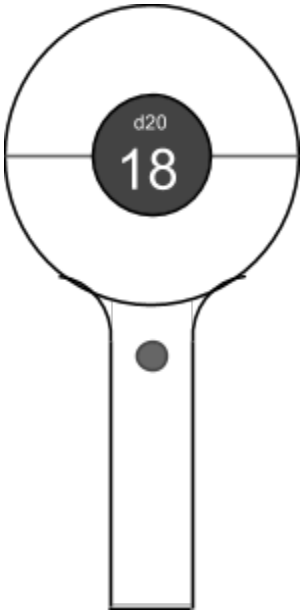



Fig. OLSM Die product concept design

# Design Exploration

## Initial Ideation

We explored a range of diverse concept designs before proceeding with the design of a universal die. We considered building the following prototypes:

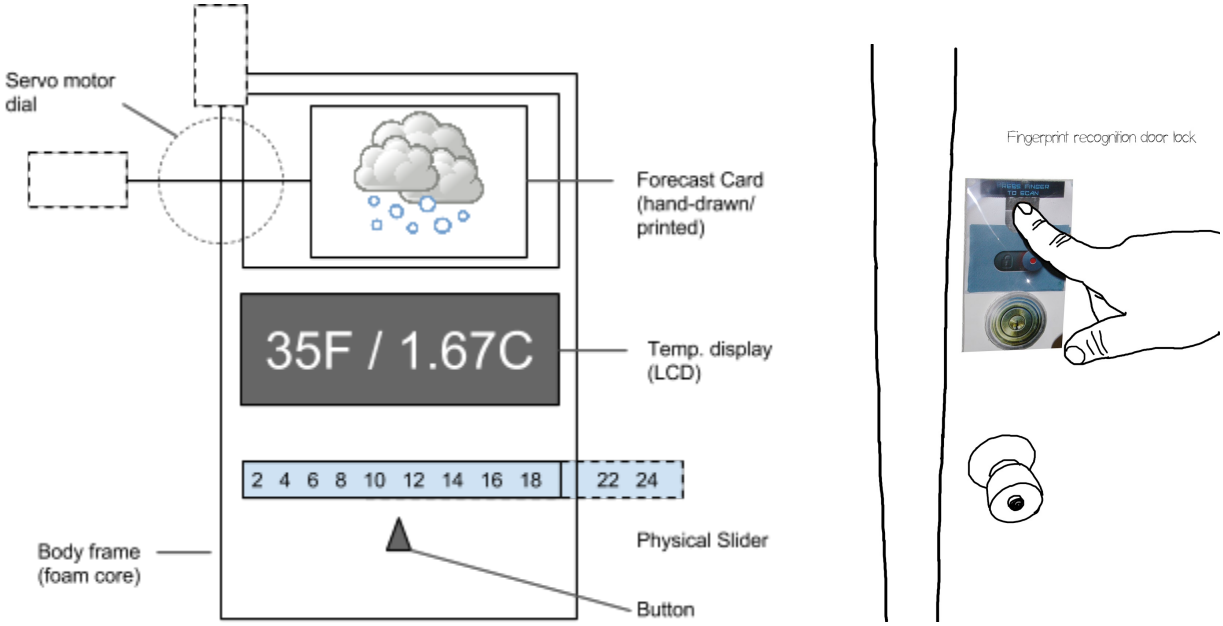



Fig. (Left) Weather display unit for quick and convenient weather information (Right) Fingerprint enabled smart lock

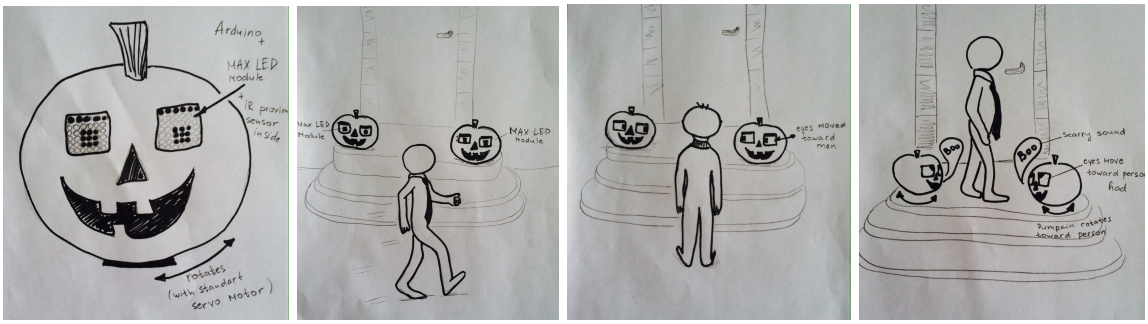
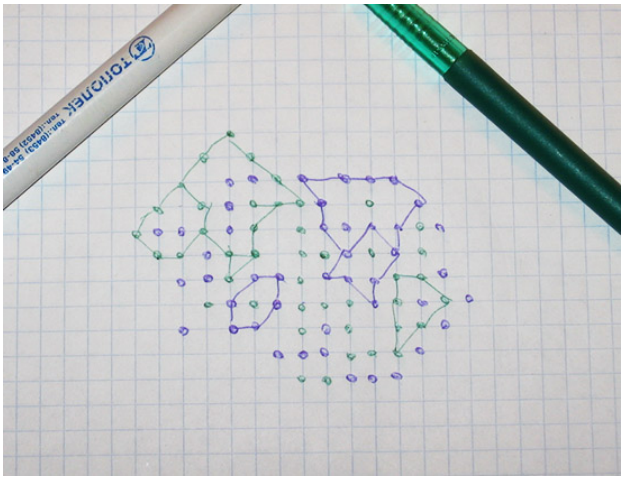



Fig. Interactive Pumpkins (proximity sensing with audio and visual feedback)



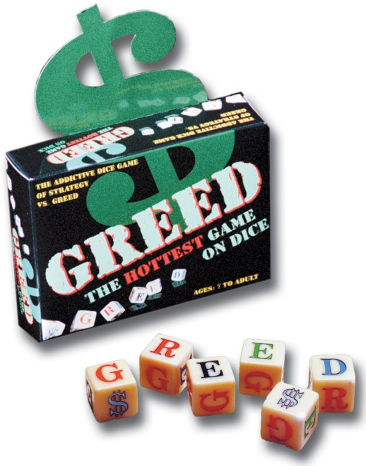



Fig. (Left) Dots game using a matrix of LEDs (Right) Greed Dice game using LCD and random number generator

## Universal Die Controller - Conceptual Design

This stage included discovering what will be the optimal design for our device. The goal was to look at many alternatives without overlooking the most optimal option. Based on a team brainstorming session and design exploration, we understood that design of the new die requires including the three pleasure principles: have aesthetic, emotional involvement and meaningful implementation. Aesthetic means that the design should be understandable and have sensory perception (Hekkert, 2006).

When we thought about the design of the universal die, we wanted to save the game factors which are present in the regular die: playfulness, sound feedback, and ease in number recognition. We also wanted to give it a completely new shape, which would be convenient to hold in hand and not easy to lose.

Additionally the shape of the OLSM die had to be big enough to include the following components:

- a button for selecting the type of die, which was located for convenience on the device's handle
- an LCD for showing information to select the type of die, on what type of die is currently selected, and what is the current roll after it has been randomly generated
- a flex sensor to recognize the shaking of the device
- a speaker for either producing sound during the shaking of the device or for feedback after the roll has been generated
- a few breadboards for housing the above components plus wires, resistors, and potentiometer
- a portable battery for power supply, which was connected to the red board

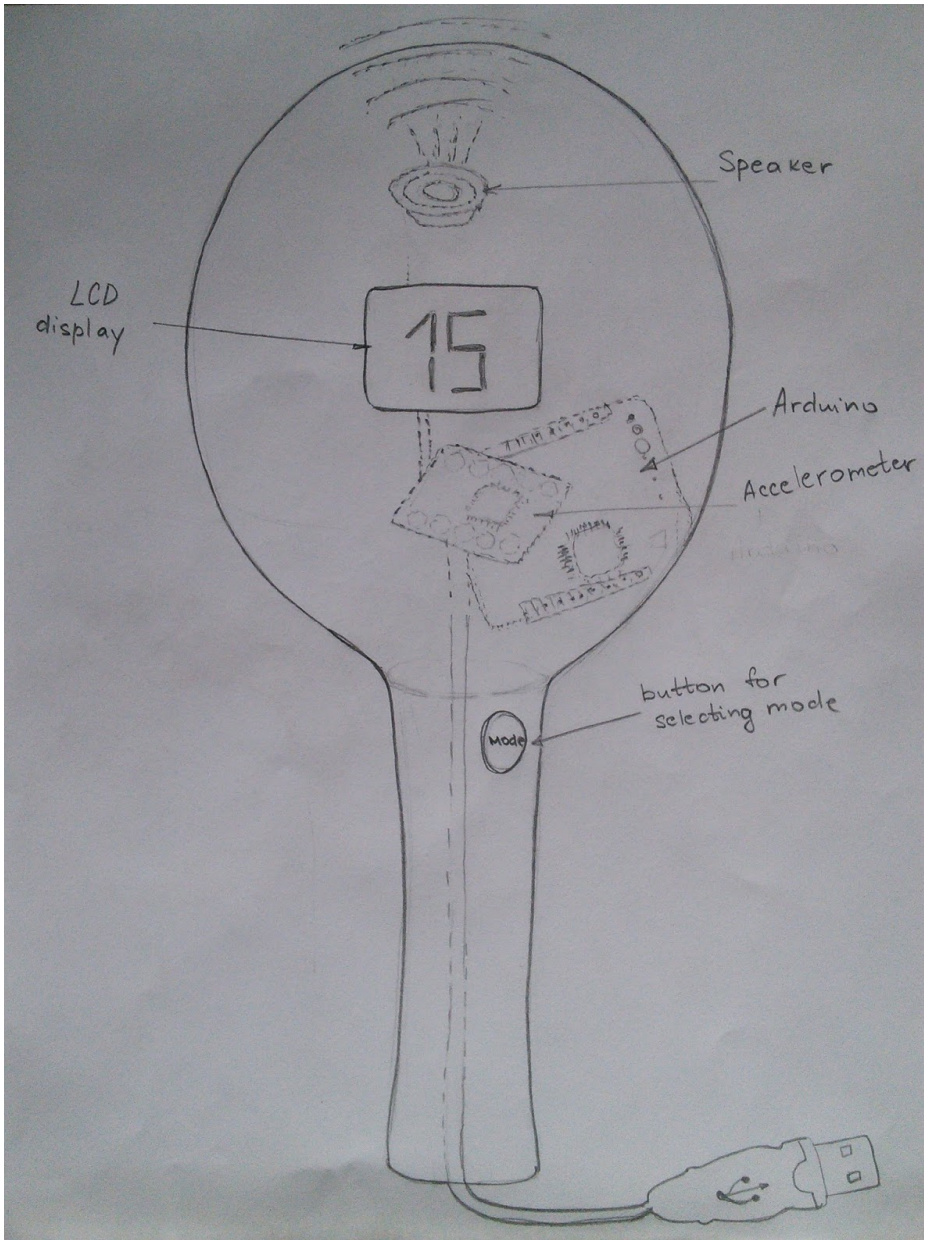



Fig. Universal die controller design sketch

Finally, we were looking at how we could evolve the user experience, improve the dice game experience and find a cost effective solution. Thus, design of our die was inspired by the shape of traditional maracas. We envisioned that such a shape will intuitively call players to shake the device and also give it the appropriate affordance.


Fig. Sketch depicting how the device may be used by players for tabletop games

# Implementation

Given that we are a four-member team, each of us decided to split the four modules (menu selection, random number generation, reading the flex sensor and playing tones/notes with the piezo speaker) amongst each member, before we got together to consolidate everything.

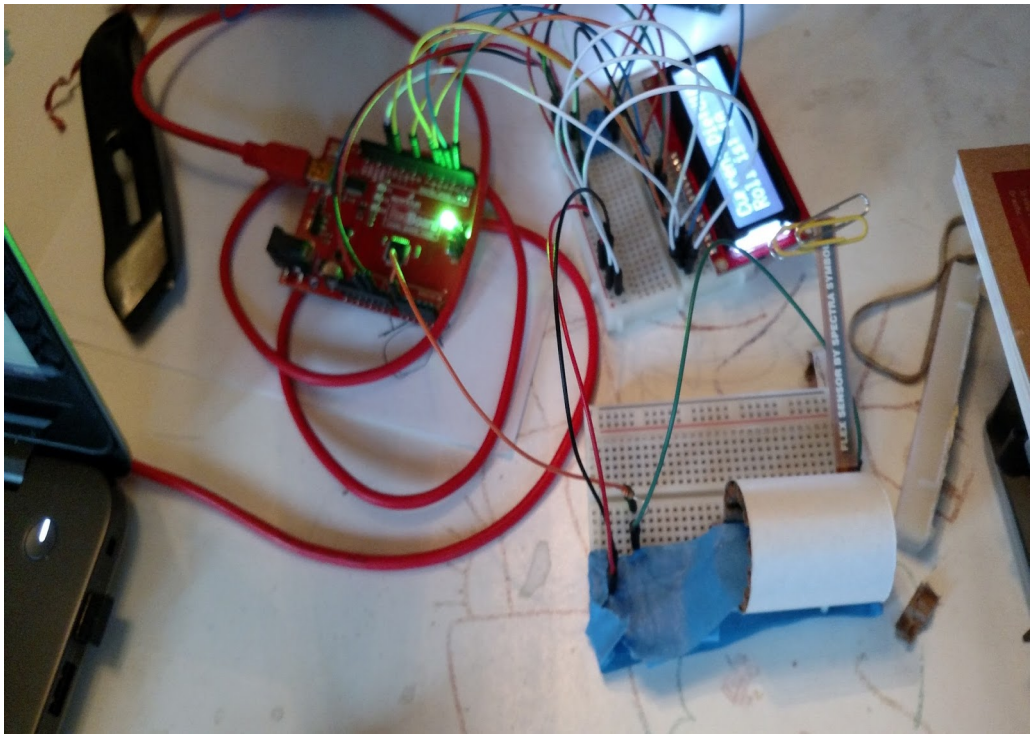


Fig. Early prototyping stage: making sure the die roll and LCD screen work

To present this prototype as a desirable alternative to rolling a set of dice, it was necessary to capture the essence of dice rolling as best as we could. Some of the things we kept in mind to mimic a die rolling experience include:

- Convenient and quick die selection
- The act of rolling the die
- Waiting for the rolled die to come to a halt before the roll can be read
- Random (unbiased) die roll when the die comes to a halt
- Feedback for the act of rolling the die and when the die comes to a halt
- Putting it all together in a form factor that makes sense to hold and play with

Each of these steps is explained in greater detail in the following sections.

## Die Selection

Die selection was implemented using a menu selection code that would allow users to choose from a d4, d6, d10 and d20 using a button that would toggle through the dice menu with each key press. Once the button was released, the die chosen would be the currently active die and would be the one that is used for a roll.

```
void Menu()
{
   lcd.clear();
   lcd.print("Current Die:");
   switch (programnumber)
   {
       case 1:  lcd.print("d4");
                upperLimit = 4;
                break;
       case 2:  lcd.print("d6");
                upperLimit = 6;
                break;
       case 3:  lcd.print("d10");
                upperLimit = 10;
                break;
       case 4:  lcd.print("d20");
                upperLimit = 20;
                break;
       default: lcd.print("d4");
                upperLimit = 4;
                break;
   }
}
```

Fig. Code snippet to choose the type of die

## Act of Rolling - Hacking the Flex Sensor

In order to facilitate shaking the device (analogous to rolling the die in one's hand) to produce a roll, we needed something like an accelerometer to detect the motion. Given the lack of components, we decided to tap into the flexibility of a flex sensor and use it as our substitute accelerometer. To do this, we first secured the base of the flex sensor such that given sufficient shaking, the lower part of the sensor would be obstructed from swinging back and forth. We then applied a slight load to the upper part of the sensor (using multiple paper clips) so that the shaking would cause the sensor to flex because of the weight at the top. This is illustrated below:

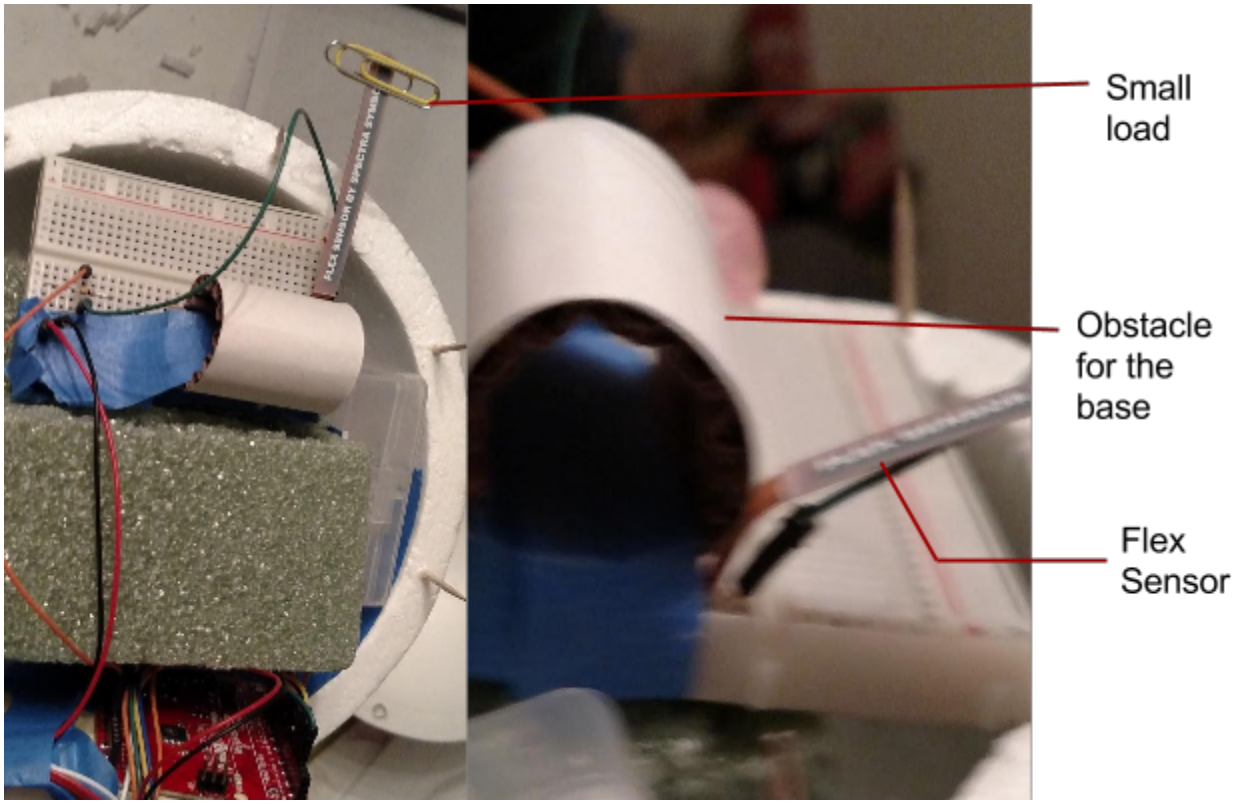



Fig. Illustrating the placement and physical configuration of the flex sensor

When in a static position, the flex sensor outputs an almost constant voltage value, that doesn't fluctuate much. With significant bending though, the voltage across the flex sensor varies, giving an output value between 0 to 1023. In our case, the value in a static position, which we will refer to as 'baseline' was about 815. To determine if the device was shook enough to be considered a roll, we check if the flex sensor outputs values less than or greater than 'baseline-20 and baseline+20'. Since, there can be a lot of fluctuation in these values based on the position of the flex sensor, we use a counter variable that must be incremented twice to be counted as a valid role. This counter is incremented each time the flex sensor throws a value outside the set tolerance range.

```
//declaration
int flexPin = 0;
unsigned int f;

//in loop()
f=analogRead(0);
baseline = 815;
//detect if the flex sensor has moved enough
if(f <= (baseline-20) || f>=(baseline+20))
        rollCnt++; //incrementing the roll count
```

Fig. Code snippet to show how to read value from the flex sensor and check the tolerance range

## Generating the Random Roll

We used the 'random(max)' function provided by the SDK, where 'max' represents the upper limit for the highest number that can be generated by the random function on the current roll. The upper limit will be determined by the current die selection.

To make the roll, we also randomized the seed being given to the pseudo random number generator, causing it to start at an arbitrary point in the random sequence. To facilitate this, we use 'randomSeed()' to initialize the random number generator with a fairly random input using analogRead on an unconnected pin. Since the analog pin is not connected to anything, the value returned by analogRead() will fluctuate based on a number of factors (e.g. the value of other analog inputs, proximity of one's hand to the board, etc.).

```
//setting up the random seed for the pseudo random generator
//initialize the pseudo-random number generator in setup()
randomSeed(analogRead(0));

//code to generate a random number. The 1 is added because
//the output of random(max) ranges from 0 to max-1.
roll = random(upperLimit) + 1;
lcd.print("Roll is: ");
lcd.print(roll);
```

Fig. Code snippet to generate and print a random number given the type of die

## Simulating the Rolling Experience

To simulate a valid roll, the program is designed in a way that it determines when the roll has been initiated and when the roll ends so that a number can be generated and displayed on the LCD.

A roll is considered as initiated when:

- a type of die has been selected (programnumber > 0)
- the button has been released (val is HIGH)
- the flex sensor output falls outside the predetermined tolerance range

A roll ends when the count exceeds 2 (rollCnt) and the die was in the act of rolling (rollStatus).

```
//for rollStatus, 0: not rolling, 1: rolling and,
//for rollEnd, 0: false, 1:true
if (val == HIGH && programnumber > 0) {
        f=analogRead(0);
        //if the flex sensor has moved enough
        if(f <= (baseline-20) || f>=(baseline+20)) {
                Shake();
                if(rollEnd == 0 && rollCnt >= 2)
                        rollEnd = 1;
        }
}

//other functions
void Shake() {
        rollCnt++;
        rollStatus = 1;
        if(rollCnt > 1 && rollStatus == 1)
                if(rollEnd == 1) {
                    Roll();
                    rollEnd = 0;
                    rollCnt = 0;
                }
}

void Roll() {
    roll = random(upperLimit) + 1;

    lcd.setCursor(0,1);
    lcd.print("Roll is: ");
    lcd.print(roll);
}
```

Fig. Code snippet to simulate rolling of a die

## Packaging it together

For prototype design, we chose two pieces of white polystyrene styrofoam sphere. Additionally, we attached a handle made out of cardboard which was glued to the sphere using Gorilla glue. We also used extra styrofoam materials to make the construction more stable. The button was placed inside the handle, while the LCD screen with the Arduino module was placed in the bottom part of the sphere. The flex sensor was fixed to the top part of the sphere so there was enough space for it to bend during the shaking of the die. Two boxes with a set of beads were placed at the sides of the sphere, to produce a maraca sound effect akin to rolling a set of dice (this was used in place of the piezo speaker). All elements inside the sphere were secured using sponge and tape.

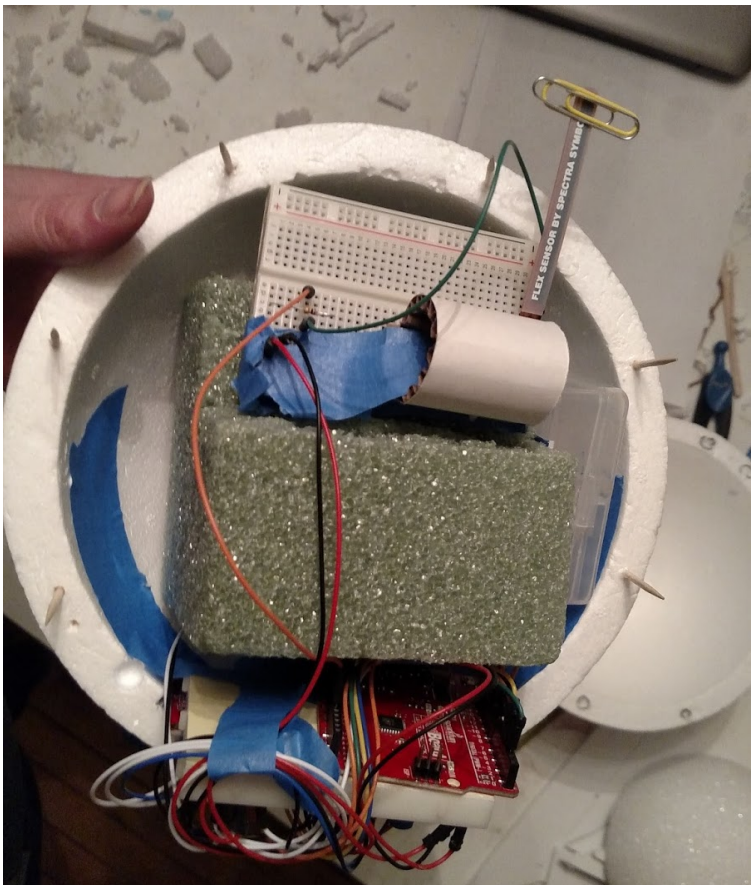



Fig. Open prototype: Entire Arduino module Uno with all the components and other supporting elements inside the styrofoam sphere

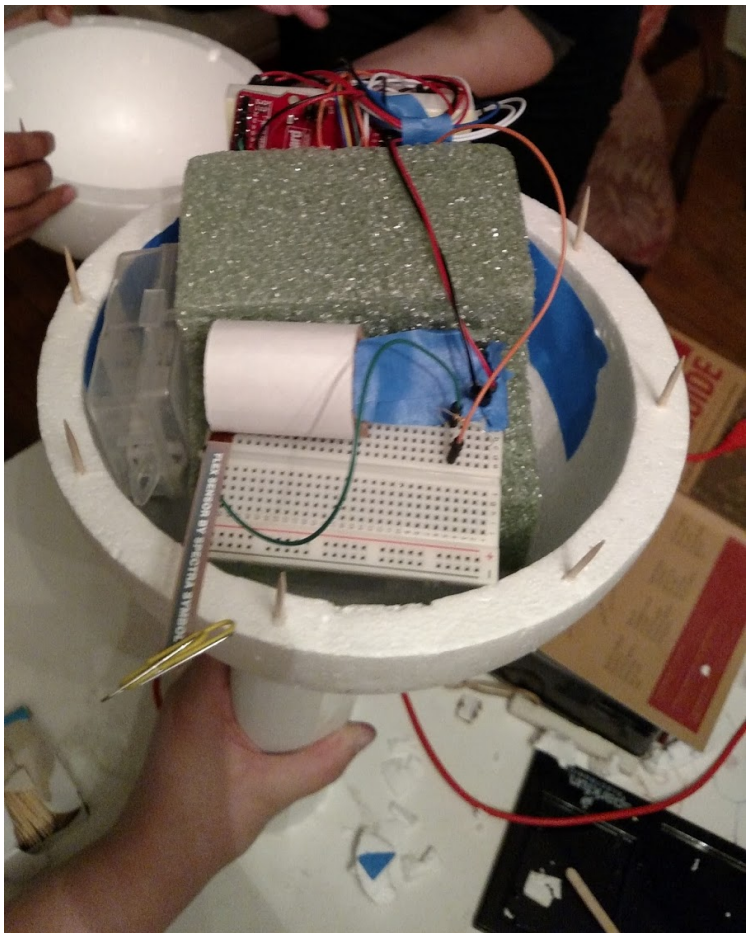


Fig. Packaging the die

To connect two parts of the sphere together, we glued regular wooden toothpicks all around the open part of the styrofoam sphere, thus, we build a construction on which we fixed the top part of the sphere.

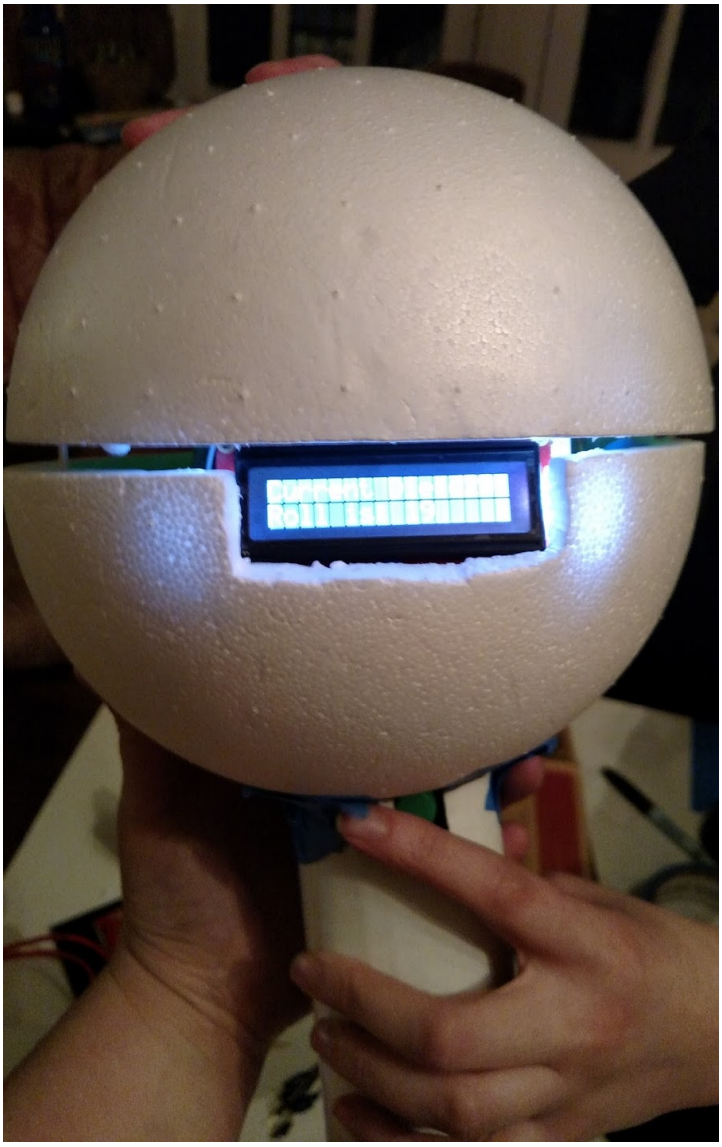


Fig. Attaching two parts of the styrofoam sphere together

*A video demo of the working prototype can be seen here:*
*https://iu.box.com/s/ml4qqelfz5budtecj0rz549uzezh1qlq*

*The complete source code can be found in this GitHub project:*
*https://github.iu.edu/saxenash/olsmDice/blob/master/sourceOLSMDice*

# Evaluation

To evaluate the OLSM Die, we presented the prototype to eight participants with varied backgrounds, including usability experts, gaming experts, tabletop gamers, and non-gamers. Overall, the response was positive - people felt the shape was playful, and could be easily decorated to match specific games and enhance their experience. One tabletop gamer also commented that they dislike the hassle of accidentally rolling the dice off the table during games, or having to keep track of several different types of dice. One particularly interesting observation we made while showing our prototype to other people was that all of our gamer participants could easily imagine using this die in a wide variety of games. Each of the gamers recommended adding the ability of customizing the look of the die to fit with different games. While we had considered this while constructing the prototype, we decided not to decorate it to avoid biasing participants towards imagining one specific use case.

The negative feedback we received was the same for all participants: the prototype is currently too large. We had assumed this would be a complaint, but the OLSM Die's size was primarily a restriction of the technology available to us. Further work on this project would yield a much smaller prototype. The other piece of negative feedback we received was from the gaming expert, who stated that the Arduino LCD screen was too small, even feeling claustrophobic. We strongly agree that the screen would need to be larger in a finalized version of the device.

Most participants readily offered suggestions for implementing the die rather than simply providing negative feedback. It very much seemed that they could imagine themselves using the device in different scenarios. One suggestion was to either add more screens, or to allow the die to project the roll on the wall to be visible by all players. We agree that adding visibility to the die roll would be necessary to prevent any kind of cheating behavior, and would implement this feature in the next prototype.

Other suggestions included adding the ability to save die rolls to refer back to later, and designing a game around the act of shaking the OLSM Die. While we had considered allowing users to roll several dice, and dice types, we had not considered the option of saving rolls. Since it is very common for players to write down their rolls during dice-reliant games, we found this to be an interesting and useful idea that we would like to implement. Finally, one participant noted the inherent tangible element involved in the physical act of rolling dice. During any one game, a single player interacts with their dice in a variety of ways, from idle fidgeting to blowing on them for good luck, that would be interesting to explore in future designs of this device.

# Lessons Learned

---

### Lessons about the assignment

- Having a good mix of skills in the team helps gain deeper insights and better solutions to problems.

- Dividing the core set of tasks amongst the team members and then putting all the pieces together helped speed up the process and gave all members ample opportunities to be a part of every phase in the overall design and implementation process.

- Bringing the group together for an intensive building/synthesizing process was incredibly beneficial to solving problems that popped up during this process.

- Variety in working methods and skills of the group members allows creating an optimal design, since each group member can bring diversity of thoughts to brainstorming ideas, describe different point of view on the idea, and point out further possible issues.

### Lessons about the device

- Most people were able to easily imagine themselves using the OLSM Die instead of a standard set of dice, implying that the form factor does NOT negatively impact perceived use cases. Players effectively saw the OLSM Die as a replacement for standard dice sets, rather than as a device created for specific types of games.

- The OLSM Die looks like fun, and makes people want to get involved.

- Tabletop gamers are NOT vehemently against the idea of using digital dice. In fact, they see this form factor as a fun, novel way to play their favorite games.

# References

Hekkert, P. (2006). *Design aesthetics: Principles of pleasure in product design.* Psychology Science, 48, 157-172.