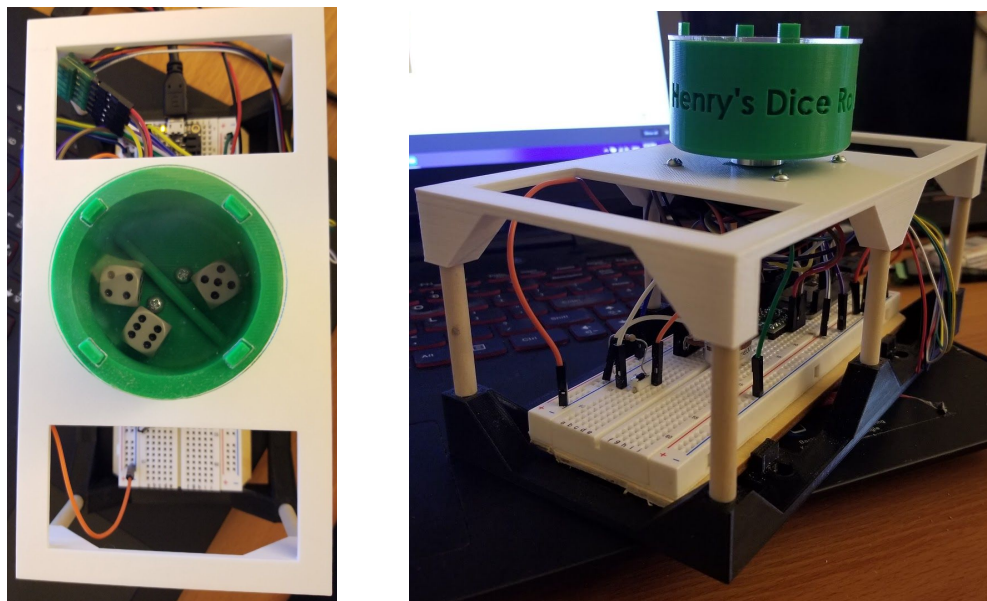


# The Automatic Dice Roller

By: Henry Cheung

## Description of the product:

My friends and I really enjoy playing Mahjong, a very popular Chinese game that involves the use of tiles in a game of strategy, memory, and skill. However, it takes a lot of time and effort to set up a game, and it accumulates after every game played. That is why I decided to create an automatic mahjong table. The project described here is a small portion of the larger project I will be working on, so this is a perfect project to do for this class. I decided to create an automatic dice roller using three dice that are used in Mahjong. The user would press a button that would automatically shuffle the dice and give them a roll result. This would contain the dice so that they do not potentially fall off the table when manually rolled, and be a cool gadget for me and my friends to use while working on the larger project. I thought that it would be a great opportunity to work on something I am already interested in and also bring in what I learned from this class into a project that I can be proud of.



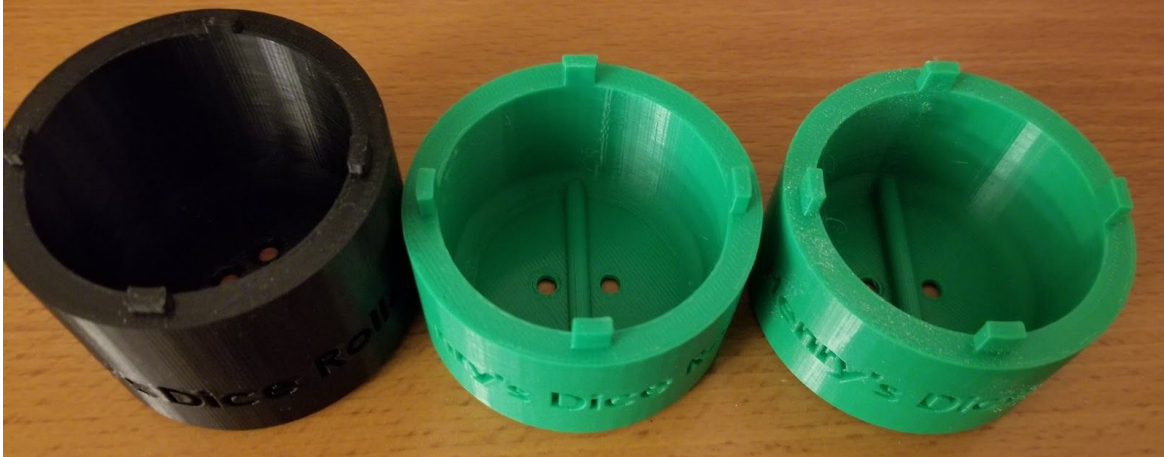
*Figure 1: My automatic dice roller mechanism, top view and isometric view*

## Electromechanical Details:

### Dice Roller Container

The main challenges in this project were in the design of the dice roller container. I first started with a basic cylindrical container design with holes at the bottom to bolt to the motor shaft hub. I used a tolerance of 0.04mm to account for the thermal expansion of the PLA. This was tuned by me after many experiences with my 3d printer. The initial design is shown as the leftmost part in Figure 2. When I tried the initial design out, I realized that the dice were spinning around as the container turned, but no rolling

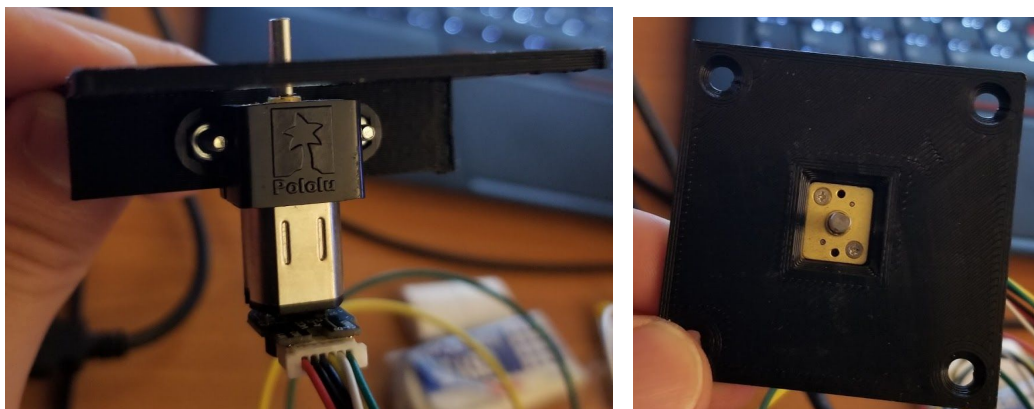
was happening. I realized that I needed something to trip the dice as they were spinning to cause them to roll, so I created the second iteration, which is the center part shown in Figure 2. This design included a hump across the diameter of the container, so that when the dice hit it while spinning, it would cause tipping over to occur. When I tested this iteration of the design out, it was working, but not as consistently as I wanted it to. As a result, I decided to experiment with the height of the hump and also whether to make the top of the hump a round semicircle or more sharp like a triangle. The final design is shown as the rightmost part in Figure 2.



*Figure 2: My dice roller container design iterations from left to right. This part sits on top of the motor shaft hub.*

### Vertical Motor Mount

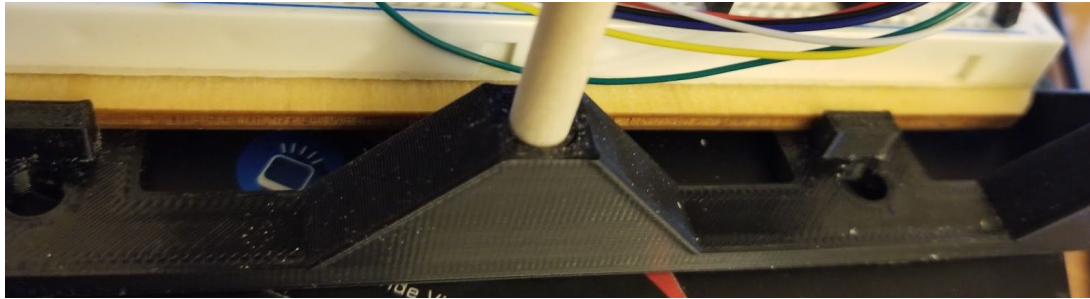
The dice roller has to be vertical in orientation to be able to get the best performance, so I also designed a motor mount for the motor I chose. This design also utilizes the plastic motor mounting bracket that came with the motor. I added in two #2 screw holes for the #2 screws used with the motor mounting bracket to be able to fix the motor to the vertical motor mount I made. Then, I had four holes on the top plane to be able to fit the vertical motor mount to the top face of the mechanism housing. I also used the same tolerances from the dice roller container here. The vertical motor mount with the motor attached can be seen in Figure 3.



*Figure 3: My vertical motor mount design. This keeps the motor in place and in an upright orientation such that the dice roller container can fit on top.*

### Mechanism Housing

I designed the mechanism housing top and bottom faces and had wooden dowels as vertical supports. This is so that the user can easily see the intricacies of the mechanism while keeping everything contained. The base part also has stubs and chamfers that are extruded to keep the breadboard from moving in the x or y direction. This can be seen in Figure 4. The top face has matching holes to mount the vertical motor mount to it.



*Figure 4: The stubs on the mechanism housing base plate and chamfers to keep the breadboard in place.*

### **Circuit:**

This project had three main elements: The microcontroller, the DC motor, and a switch.

1. **Microcontroller:** I used the HUZAZH32 Feather Microcontroller. This was the microcontroller that was given to me in the lab kit to use. For my application, this controller was more than enough to do the job because it could send PWM commands to the motor, generate timeframes for the duration of motor spin, and power the button.
2. **DC Motor:** I used the 75 : 1 Micro Metal Gearmotor HP 6V with Extended Motor Shaft. This motor was also given to me in the lab kit to use. DC motors are the correct application for this because I do not require precision or high load. This motor is a brushed DC motor that uses a stator and a rotor, and it comes with a gearbox attached to the output shaft.
3. **Switch:** I used the switch given in the lab kit to trigger the mechanism to start. To select which pull-up resistor size is appropriate, the pull-up resistor should be an order of magnitude (1/10th) less than the input impedance of the pin. The input impedance of the pin is inferred to be around 100k $\Omega$ , so the constraint would be:  $R=100 * 0.1 =10k\Omega$ . Once I had that value, I checked to see if the limiting current would be with that value as the pull-up resistance value. Since 0.00033A is close to 0, 10k $\Omega$  is a good choice for this circuit.

The wiring diagram for this mechanism is shown below in Figure 5.

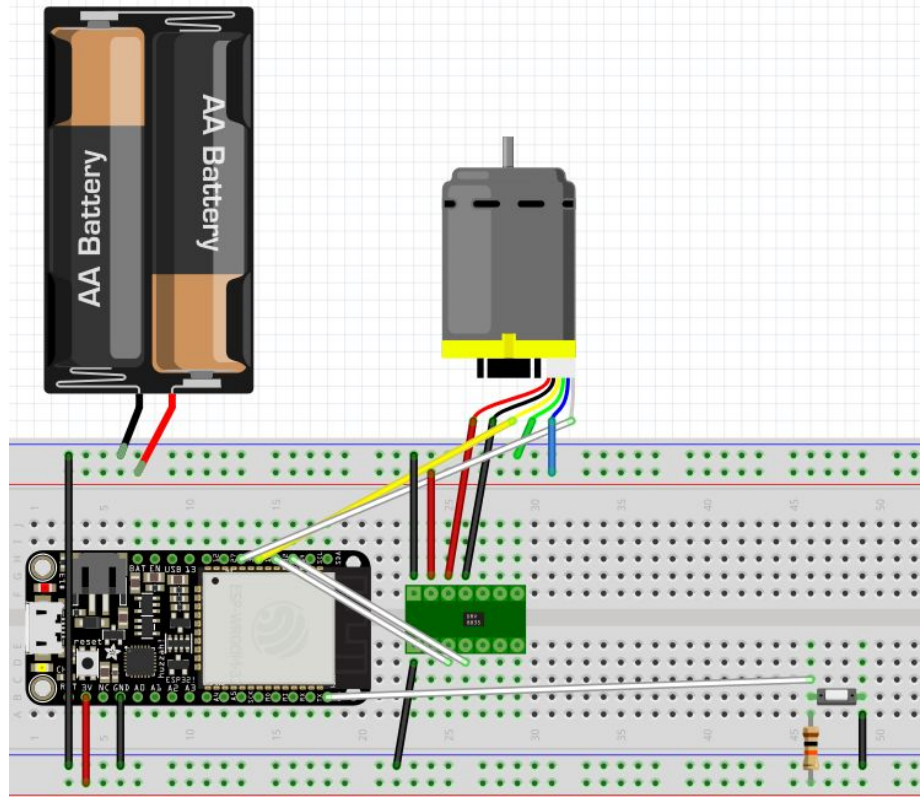
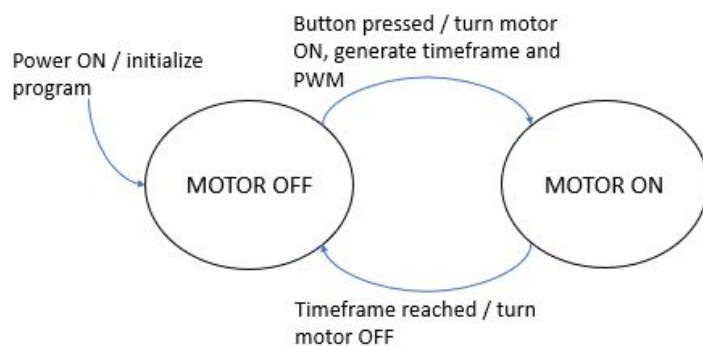


Figure 5: Wiring diagram for the automatic dice roller mechanism.

### Finite State Machine:

The behavior of this machine is quite simple, since the user presses the button to begin the dice roll, and the micro will designate how long the rolling goes on for using a random number generator from 1 to 5 seconds and at what PWM to command the motor using a random number generator from a lower limit PWM to 255. The lower limit was found using manual testing and seeing the performance of the dice shuffling to find the lowest PWM that would do the job. In this case, the lower limit was a PWM of 200. Once the motor runs for the allotted time set by the microcontroller, the motor stops and the resulting dice roll value is revealed.



## Appendix I: Arduino Code

```
//Henry Cheung 3032663584
//ME102B Project: Dice Roller

#define FREQ 20000 //define PWM frequency
#define Bin1 32 // define motor driver Bin1 pin to provide logic to choose direction of DC motor
#define Bin2 14 // define motor driver Bin2 pin to provide logic to choose direction of DC motor
#define button 21 //define button digital pin
volatile bool interrupt = false; //initialize interrupt boolean as false
volatile bool finished = false; //initialize boolean to stop roll
bool generated = false; //initialize boolean to signify whether rand num has been generated
float dutyCycle = 0; //initialize duty cycle variable
float timeEnd = 0; //initialize variable for end time
float timeFrame = 0; //initialize variable for timeframe
float t = 0; //time elapsed

void IRAM_ATTR button_interrupt() { //ISR function for button
  interrupt = true; //set interrupt to true
}

void setup() {
  Serial.begin(115200); //setup serial reading
  pinMode(Bin1, OUTPUT); //set logic pin Bin1 to output
  pinMode(Bin2, OUTPUT); //set logic pin Bin2 to output
  pinMode(button, INPUT); //set button pin to input
  attachInterrupt(digitalPinToInterrupt(button), button_interrupt, RISING); //attach ISR to button
  ledcAttachPin(Bin1, 1); //assign Bin1 to channel 1
  ledcAttachPin(Bin2, 2); //assign Bin2 to channel 2
  ledcSetup(1, FREQ, 8); //12kHz PWM, 8-bit resolution
  ledcSetup(2, FREQ, 8); //12kHz PWM, 8-bit resolution
}

void loop() {
  switch (interrupt) { //checking button state
    case true: //button pressed
      if (!generated) { //random number not generated yet
        timeFrame = random(1,6) * 1000; //generate timeframe
        t = millis() + timeFrame; //set t to the program time when motor stops
        Serial.print("Timeframe: "); //print the timeframe
        Serial.print(timeFrame);
        Serial.println(" ms");
        dutyCycle = random(200, 255); //generate duty cycle from lower limit to max
        Serial.print("duty cycle: "); //print the duty cycle
        Serial.println(dutyCycle);
        generated = true; //set generated to true
      }
    }
  }
}
```

```
if (millis() >= t) { //if timeframe is reached
    finished = true; //set finished to true
}
switch (finished) { //check state of finished
    case true: //finished is true
        ledcWrite(1, 0); //set channel 1 to 0
        ledcWrite(2, 0); //set channel 2 to 0
        interrupt = false; //set interrupt to false
        finished = false; //set finished to false
        generated = false; //set generated to false
        break;
    case false: //finished is false
        ledcWrite(1, dutyCycle); //set channel 1 to High
        ledcWrite(2, 0); //set channel 2 to 0
        break;
}
break;
case false: //button not pressed
    ledcWrite(1, 0); //set channel 1 to 0
    ledcWrite(2, 0); //set channel 2 to motor_speed
    break;
}
}
```