TO: EGR Corporation
FROM: Izzy Dudlyke, Juan Garcia Luego, Manya Kaushik
SUBJECT: Sportbot Machine
DATE: April 15, 2022

Executive Summary

After discussion with the wider group and deciding on the theme of Candyland, the team began the brainstorming process to reach a SportBot design. Our SportBot would be the Candyland castle and would consist of a rising drawbridge to move the ball and a light show triggered by the ball when it moves. The electronics component utilized a 555 timer circuit in order to rotate a motor for approximately 4 seconds in order to make the bridge rise. The arduino component consisted of an ultrasonic sensor which would trigger an LED strp when the ball was near. Through multiple iterations of the circuits, the team chose to use a switch rather than the initial idea of a photoresistor for the electronics circuit as it was more reliable, and individual LEDs instead of an LED strip. The SportBot was then constructed and assembled, integrating the electronic components into the mechanical design. Our SportBot was extremely reliable and would consistently function correctly when the ball was placed in the drawbridge. The overall group worked together extremely well, with members of different teams willing to be proactive and help other teams if they were having problems. We were able to successfully problem solve and work through challenges that arose in assembly in order to have a functional ass that was able to cohesively complete two loops through all 5 SportBots.

Approach Description and Results

I. Key Design Decisions and Outcomes

After consulting with the larger group and deciding on the theme of Candyland, the team began to brainstorm ideas for their individual machine. Through brainstorming and researching the different aspects of Candyland, the team chose the Candyland Castle as their individual component. The chosen design was a drawbridge that when lifted, rolls a ball through the Candyland Castle, to the next component in the Rube-Goldberg Machine. The electronic component was designated to be the drawbridge, as the team knew that a 555 timer could be utilized to bring the drawbridge up with a motor for a certain amount of time. The team initially chose a photoresistor as the input for the timed motor, meaning that when the photoresistor got covered by the ball, the motor would be triggered and would bring the ball down the drawbridge. For the arduino component, the team initially decided to use led strips for a light show around the castle, using an ultrasonic sensor as the input. These were the initial design ideas that the team decided on.

When the team began actually designing and making the machine, there were some changes made to adjust to challenges. First, the team worked on making the circuit components.

For the electrical circuit with the 555 timer, motor, and photoresistor, the team found that the circuit did not function correctly. After trying many times to get this circuit to work to no avail, the team instead decided to use a push button switch as the input for the motor (Figure 2). This worked successfully, and was a good decision in the end as it was more consistent than the use of the photoresistor. It did not affect the overall system too much, as they had not designed the structure yet, so they could adapt to the new input trigger. The team then designed the drawbridge to use a wind up motion driven by a motor when the push button was triggered.

Another problem arose when the team attempted to create the LED strip light show around the castle. When trying to create the code with the ultrasonic sensor and the LED strip, the team found that the LED strip was confusing to use and non-functional. Therefore, the team instead decided to use six singular LED's that would be placed around the rim of the castle. The LED's were not placed on the drawbridge itself, as this would weigh down the drawbridge and potentially make it difficult for the motor to wind it up. The final outcome for the arduino aspect was six leds that lit up in succession around the castle when the ball triggered the ultrasonic sensor (Figure 3, Appendix Figure 2).

As the motor was to be within the box structure to wind up the motor, the team attached a wheel to the motor, which was easily accessible from the outside of the box in order to easily reset the bot to its starting position (Appendix Figure 3). This was extremely successful and useful, as it meant once the electronic components were in place, we never had to move any to unwind the motor and reset. This significantly reduced any damage to our SportBot and allowed it to consistently work.

II. Sportbot Action Performance Summary

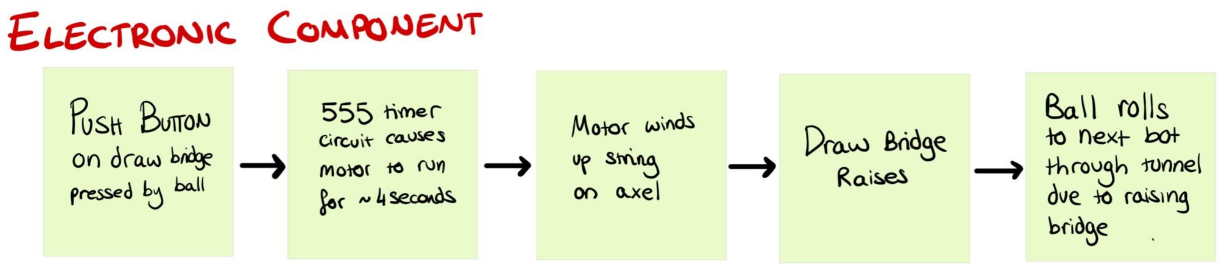

ELECTRONIC COMPONENT
Push Button on draw bridge pressed by ball
555 timer circuit causes motor to run for ~4 seconds
Motor winds up string on axel
Draw Bridge Raises
Ball rolls to next bot through tunnel due to raising bridge

Figure 3a: Electronics Component Action Performance Summary

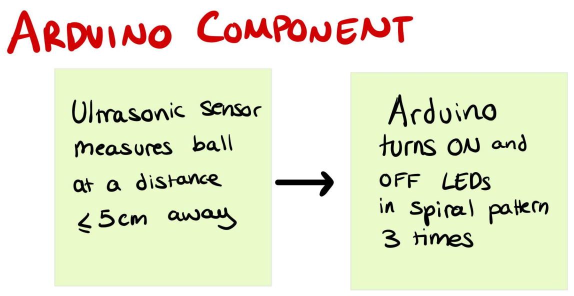

ARDUINO COMPONENT
Ultrasonic sensor measures ball at a distance ≤5cm away
Arduino turns ON and OFF LEDs in spiral pattern 3 times

Figure 3b: Arduino Component Action Performance Summary

After constructing a working prototype with the drawbridge attached to the motor and the ultrasonic sensor, we did not encounter any flaws with the design. The electronic circuit and arduino part worked consistently. The ultrasonic sensor was placed so that the ball was sensed in every trial.

When integrating our prototype with the rest of the DC6 groups, we did not have to change any component of our design. We added a bridge between the exit point of the ball in the castle and the entry point in the Gingerbread House.

During testing of the joined Group A DC6 projects, the team's prototype only failed when the exit bridge was not aligned with the path of the ball. Both the electronic circuit and arduino remained unchanged throughout the entire testing phase as they worked as planned. Ultimately, in the final test, the team's bot worked exactly as designed and no problems were encountered on our machine throughout the 30 test minutes.

III. Electronic schematic

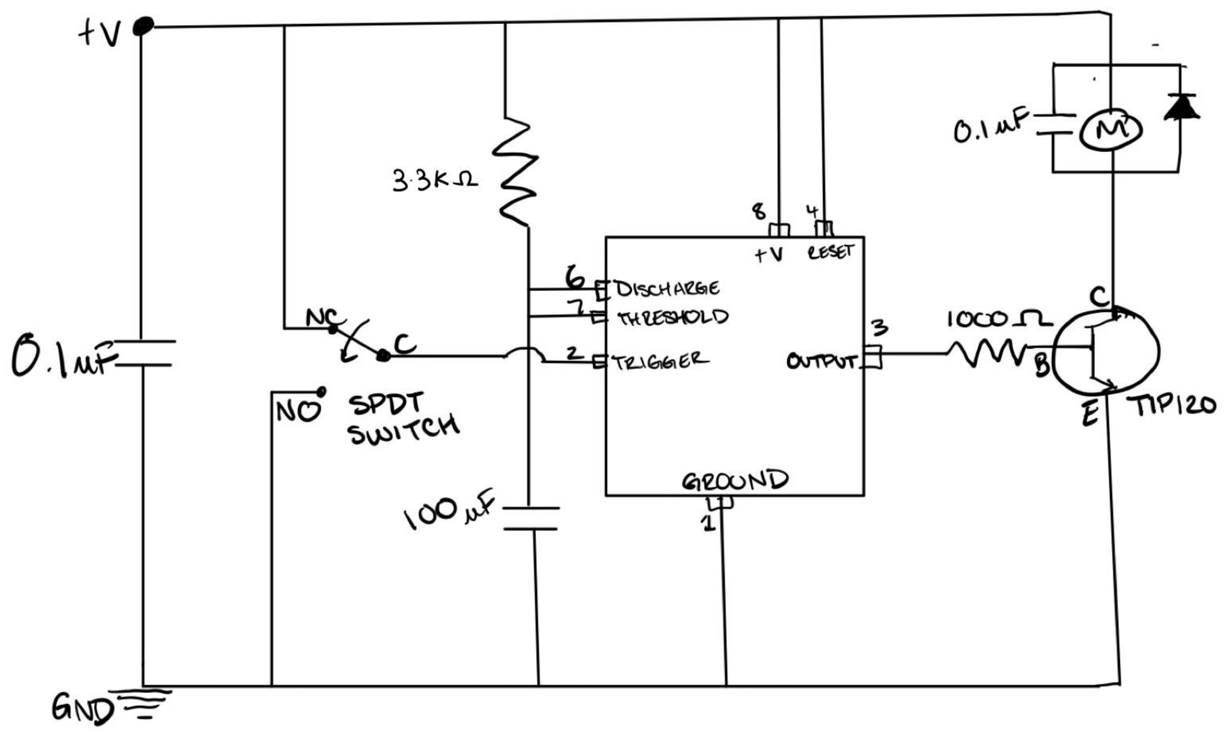


Figure 2: Circuit Schematic for Electronic Component (Push Button Timed Motor)

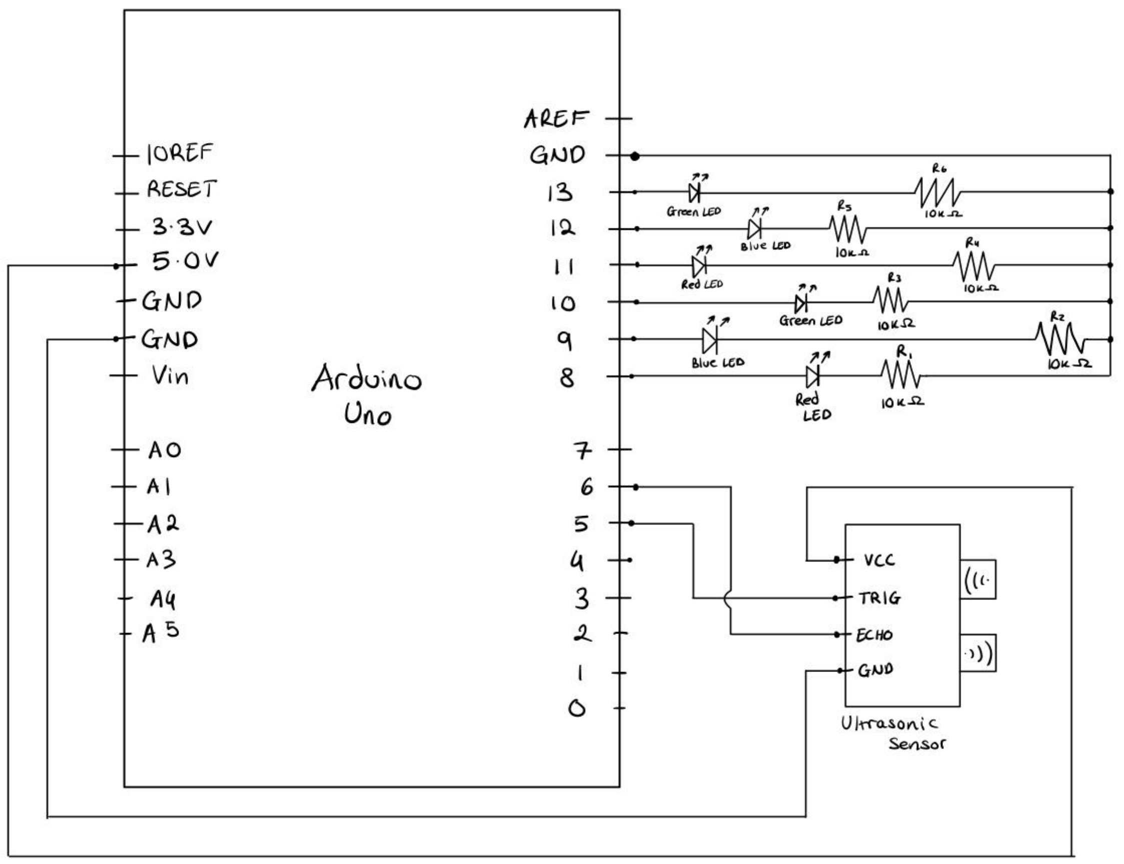


Figure 3: Circuit Schematic for Arduino Component (LED and Ultrasonic Sensor)

IV. Personal contribution summary

Izzy	**Manya**	**Juan**
- 3D printed Castle - Constructed and assembled the bot with the electronic components. - Created and soldered the LED circuit - Aesthetics/Design of the entire structure (Pom-pom details)	- Created and soldered electronics circuit for the drawbridge component - Worked on assembling the entire machine (combining electrical and mechanical components)	- Created Arduino code for the LED light show on LEDs - Worked on LED strip design - Helped with integration with the rest of Group A teams.

V. Commentary on overall group contribution

The overall group worked very well together, as everyone was proactive and communicative with each other throughout the process. The group began by coordinating a theme together and choosing the different machines for each individual team to work on. Additionally, the overall group discussed dimensions before the individual teams started creating their machines, which was helpful and something that worked well. This ensured that all of the individual machines fit within the 4' by 4' box that was designated. One problem that the group had to solve was the height dependency of the overall Rube-Goldberg Machine. Each of the machines had a different necessary height for the previous and successive bots, which was difficult to coordinate. Although initially discussing heights as a group to map this out, during construction many teams found their individual sport bots had to change dimensions. This meant that the heights of each machine had to be adjusted when the machines were finally put together, which was a challenge. However, after many rounds of testing and alterations, this was solved by increasing the height of the first machine, and having the collection back to the first bot at a lower level. As the group initially planned on a much lower overall height, this increased height still lay within the maximum 4 foot height.

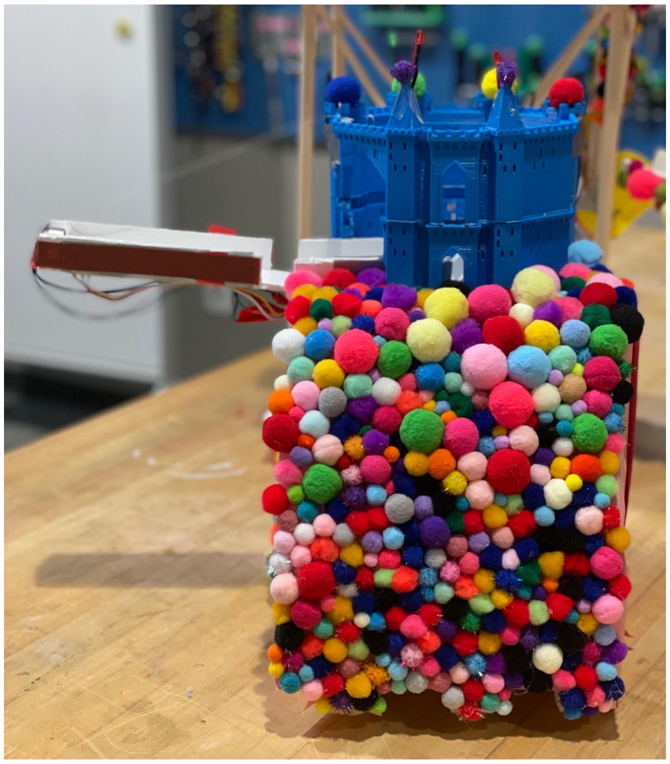
Another problem that arose the day before performance of the machine, was that individual bots began to malfunction. On the day before the performance, two of the five bots had problems, but the teams were able to work together to try to fix them. The overall group was helpful and everyone worked together to try and fix the problems. This worked very well in our group and made it easier to fix the problems. The final bot did still malfunction on the day of the performance, but this was not too much of a problem because the ball was still able to complete two loops through the machines.

In order to make decisions as a group, the main technique used was brainstorming as a group and voting on a solution. When there were problems with incorporating all the machines together, such as height problems, members would make suggestions about what could potentially help, and then as a group we would test these solutions to see what worked best. Although this ended up taking a long time, it was successful in solving the problems. Something that the team would do differently in terms of the group would be to start incorporating all the machines together earlier in the process. The group met to do this two days before and the day before the performance, but the team would like to start earlier if they were to do this again. The incorporation of the machines together took longer and was more tedious than the group anticipated, which is why having more time to do this would have been beneficial. Besides this, the overall group worked well together, as everyone communicated and helped each other with their machines to allow the entire Rube-Goldberg Machine to succeed.

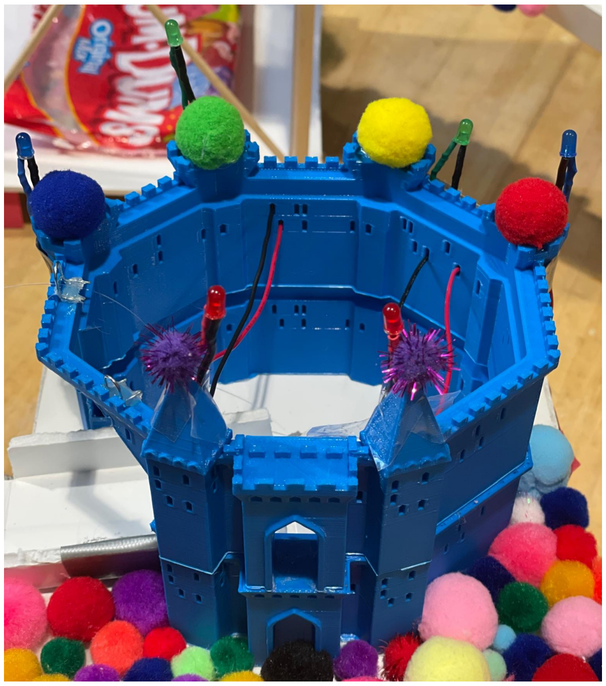
Conclusions

Although both as an individual team and as a larger group, we encountered many challenges in building and integrating the SportBots, we were able to effectively cooperate as a team in order to create a working and cohesive Rube-Goldberg machine. The overall theme was visually exciting and colorful, which made the loop even more entertaining to watch. The Candyland Castle SportBot worked consistently in every trial which ultimately helped the entire team succeed in the Design Challenge 6.

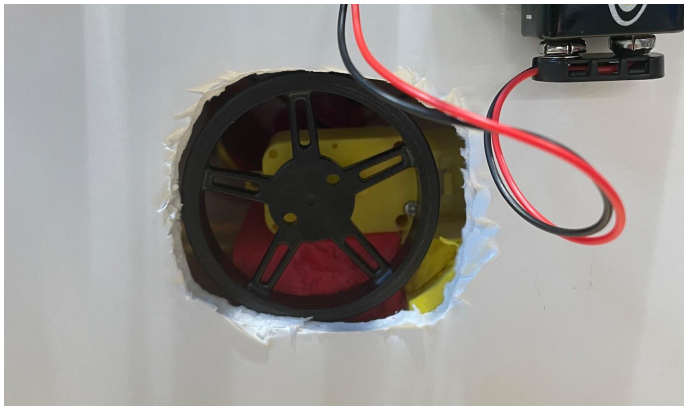
Appendix

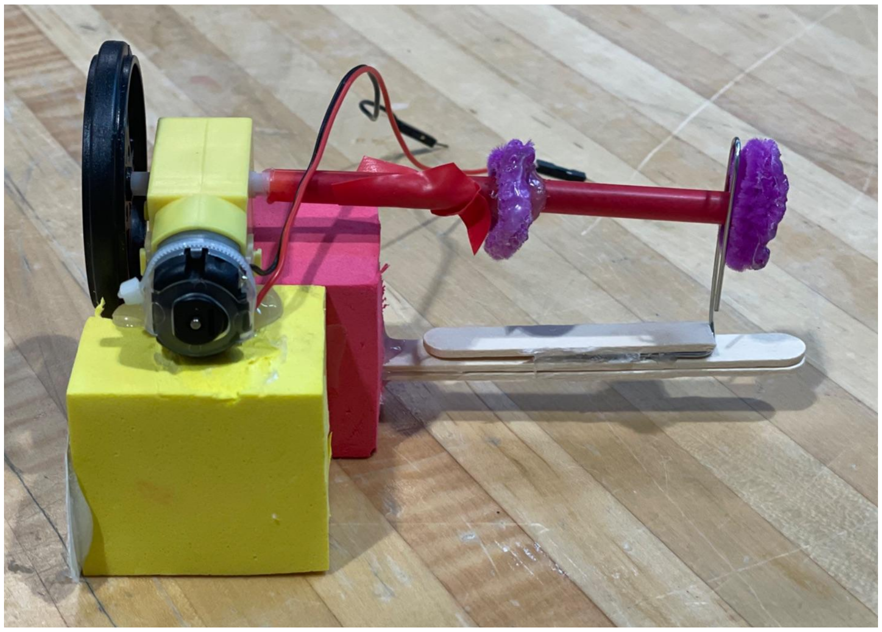


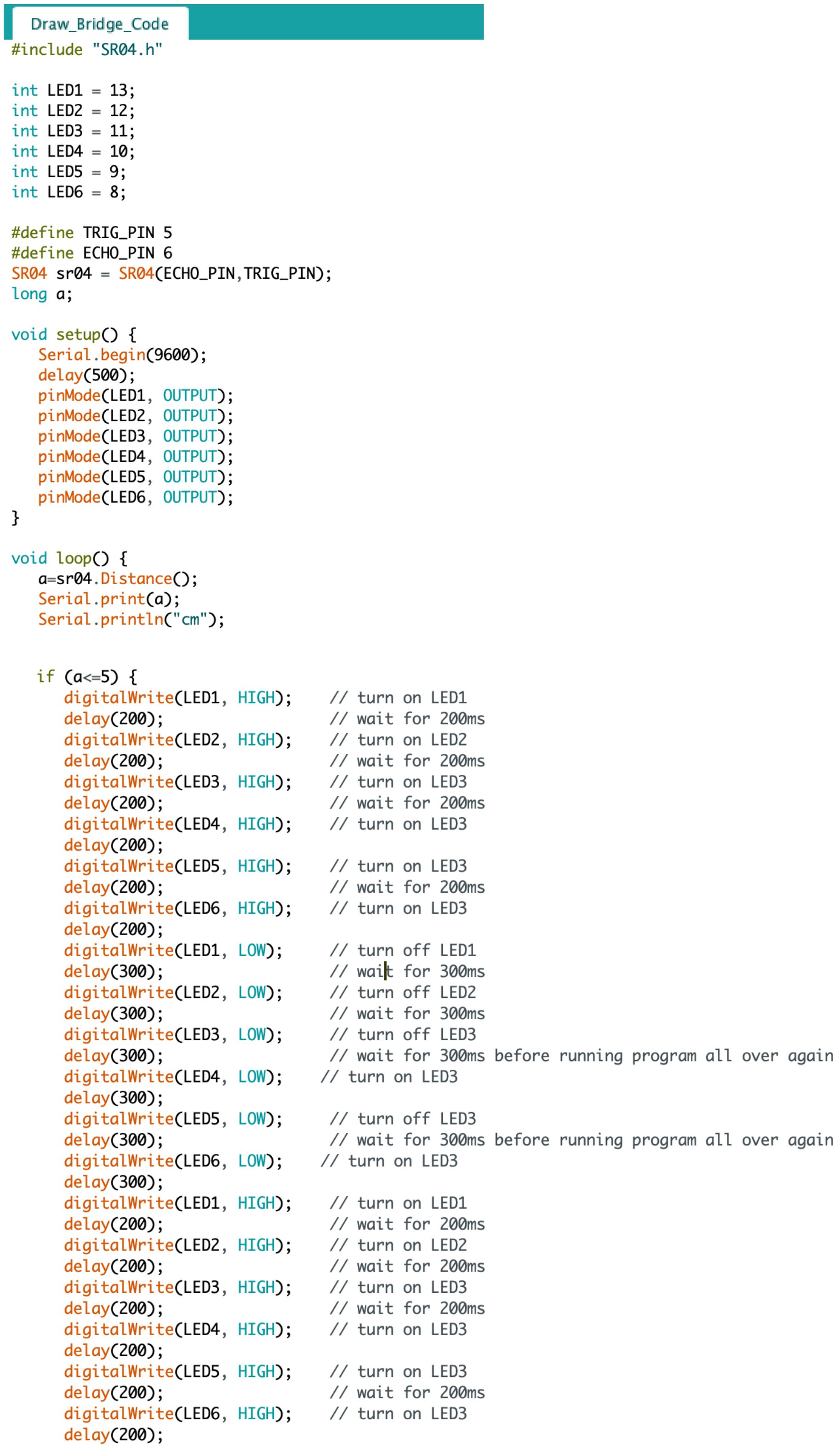
Appendix Figure 1: Candyland Castle Structure with gumball waterfall



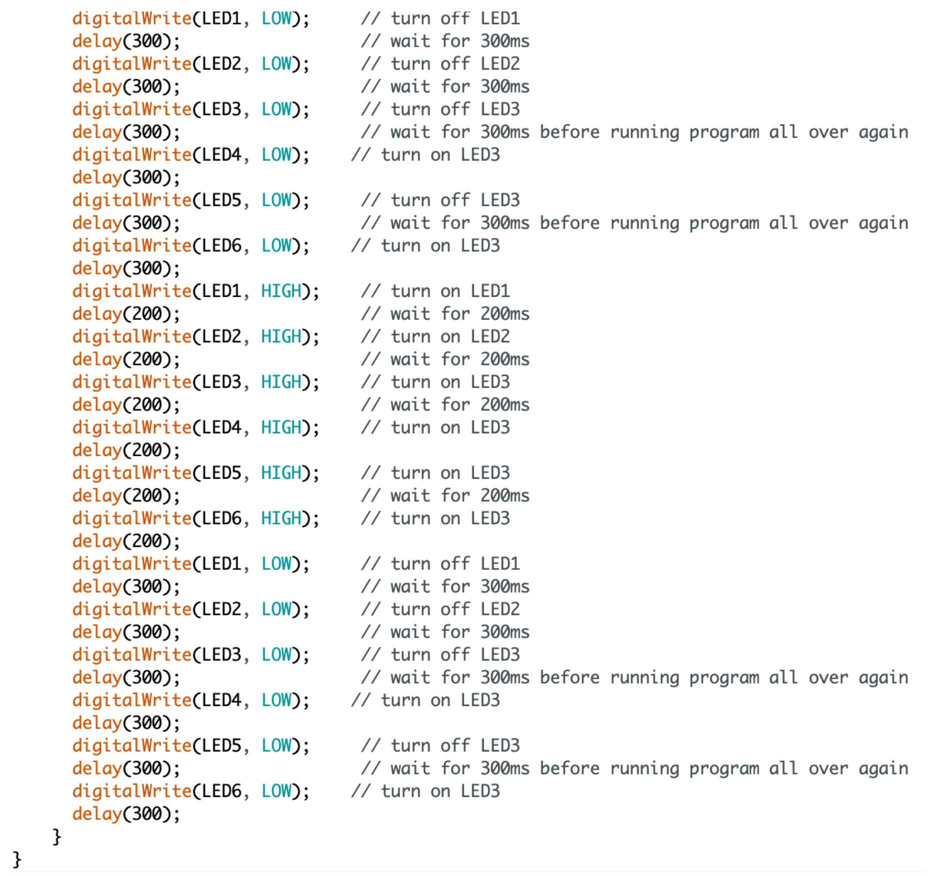
Appendix Figure 2: Arduino Controlled LEDs on Candyland Castle


Appendix Figure 3: Easily accessible wheel to unwind motor and reset bot


Appendix Figure 4: Motor Stand with axle and stabilization method to stop axle from bending


Draw_Bridge_Code

```
#include "SR04.h"

int LED1 = 13;
int LED2 = 12;
int LED3 = 11;
int LED4 = 10;
int LED5 = 9;
int LED6 = 8;

#define TRIG_PIN 5
#define ECHO_PIN 6
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
long a;

void setup() {
   Serial.begin(9600);
   delay(500);
   pinMode(LED1, OUTPUT);
   pinMode(LED2, OUTPUT);
   pinMode(LED3, OUTPUT);
   pinMode(LED4, OUTPUT);
   pinMode(LED5, OUTPUT);
   pinMode(LED6, OUTPUT);
}

void loop() {
   a=sr04.Distance();
   Serial.print(a);
   Serial.println("cm");


   if (a<=5) {
      digitalWrite(LED1, HIGH);    // turn on LED1
      delay(200);                  // wait for 200ms
      digitalWrite(LED2, HIGH);    // turn on LED2
      delay(200);                  // wait for 200ms
      digitalWrite(LED3, HIGH);    // turn on LED3
      delay(200);                  // wait for 200ms
      digitalWrite(LED4, HIGH);    // turn on LED3
      delay(200);
      digitalWrite(LED5, HIGH);    // turn on LED3
      delay(200);                  // wait for 200ms
      digitalWrite(LED6, HIGH);    // turn on LED3
      delay(200);
      digitalWrite(LED1, LOW);     // turn off LED1
      delay(300);                  // wait for 300ms
      digitalWrite(LED2, LOW);     // turn off LED2
      delay(300);                  // wait for 300ms
      digitalWrite(LED3, LOW);     // turn off LED3
      delay(300);                  // wait for 300ms before running program all over again
      digitalWrite(LED4, LOW);    // turn on LED3
      delay(300);
      digitalWrite(LED5, LOW);     // turn off LED3
      delay(300);                  // wait for 300ms before running program all over again
      digitalWrite(LED6, LOW);    // turn on LED3
      delay(300);
      digitalWrite(LED1, HIGH);    // turn on LED1
      delay(200);                  // wait for 200ms
      digitalWrite(LED2, HIGH);    // turn on LED2
      delay(200);                  // wait for 200ms
      digitalWrite(LED3, HIGH);    // turn on LED3
      delay(200);                  // wait for 200ms
      digitalWrite(LED4, HIGH);    // turn on LED3
      delay(200);
      digitalWrite(LED5, HIGH);    // turn on LED3
      delay(200);                  // wait for 200ms
      digitalWrite(LED6, HIGH);    // turn on LED3
      delay(200);
```



```
      digitalWrite(LED1, LOW);     // turn off LED1
      delay(300);                  // wait for 300ms
      digitalWrite(LED2, LOW);     // turn off LED2
      delay(300);                  // wait for 300ms
      digitalWrite(LED3, LOW);     // turn off LED3
      delay(300);                  // wait for 300ms before running program all over again
      digitalWrite(LED4, LOW);    // turn on LED3
      delay(300);
      digitalWrite(LED5, LOW);     // turn off LED3
      delay(300);                  // wait for 300ms before running program all over again
      digitalWrite(LED6, LOW);    // turn on LED3
      delay(300);
      digitalWrite(LED1, HIGH);    // turn on LED1
      delay(200);                  // wait for 200ms
      digitalWrite(LED2, HIGH);    // turn on LED2
      delay(200);                  // wait for 200ms
      digitalWrite(LED3, HIGH);    // turn on LED3
      delay(200);                  // wait for 200ms
      digitalWrite(LED4, HIGH);    // turn on LED3
      delay(200);
      digitalWrite(LED5, HIGH);    // turn on LED3
      delay(200);                  // wait for 200ms
      digitalWrite(LED6, HIGH);    // turn on LED3
      delay(200);
      digitalWrite(LED1, LOW);     // turn off LED1
      delay(300);                  // wait for 300ms
      digitalWrite(LED2, LOW);     // turn off LED2
      delay(300);                  // wait for 300ms
      digitalWrite(LED3, LOW);     // turn off LED3
      delay(300);                  // wait for 300ms before running program all over again
      digitalWrite(LED4, LOW);    // turn on LED3
      delay(300);
      digitalWrite(LED5, LOW);     // turn off LED3
      delay(300);                  // wait for 300ms before running program all over again
      digitalWrite(LED6, LOW);    // turn on LED3
      delay(300);
    }
}
```

Appendix Figure 5: Arduino Code for LED Lights