Report: Ravioli Machine

Group 9: Ian Zhang, Annette Bennet, Marissa Shoji

Opportunity

Ravioli is a common popular dish; however, current methods of making it involve a lot of time and effort, which may pose problems for disabled people, or those who are too busy. The goal of our project is to automate the ravioli making process, making it faster and easier. Rather than requiring an individual to press and cut each individual ravioli, similar to pasta machines, our project continuously presses two sheets of premade dough together around filling fed into a hopper at the top, automatically shaping and cutting the ravioli at the press of a button. This allows for a lot of ravioli to be made rapidly with minimal effort.

High-level Strategy

Our ravioli maker consists of two rollers that press sheets of dough together around the filling, which is fed in through the hopper at the top. The two sheets of dough are fed in through the sides of the machine in the gap between the hopper and the casing, and the rollers shape and cut the ravioli. The machine stops when the filling runs out, or when the user pushes the button. The motor turns the rollers with enough torque to simultaneously cut through the dough and draw more of it into the machine on its own, with little help from the user. The roller speed is adjusted by turning the potentiometer. This allows a large batch of ravioli to be shaped and cut, which emerges from the bottom of the machine as a sheet of ravioli that tears apart easily along the cut lines. The machine is easily disassembled for cleaning, and the motor module can be removed and replaced with a hand crank, if the user chooses to crank the ravioli maker by hand.

Our initial goals also included having the ravioli maker stop if the dough or the filling runs out, and having the machine load the dough and filling itself; of these, we were able to accomplish having the machine stop if it senses that the filling has run out.

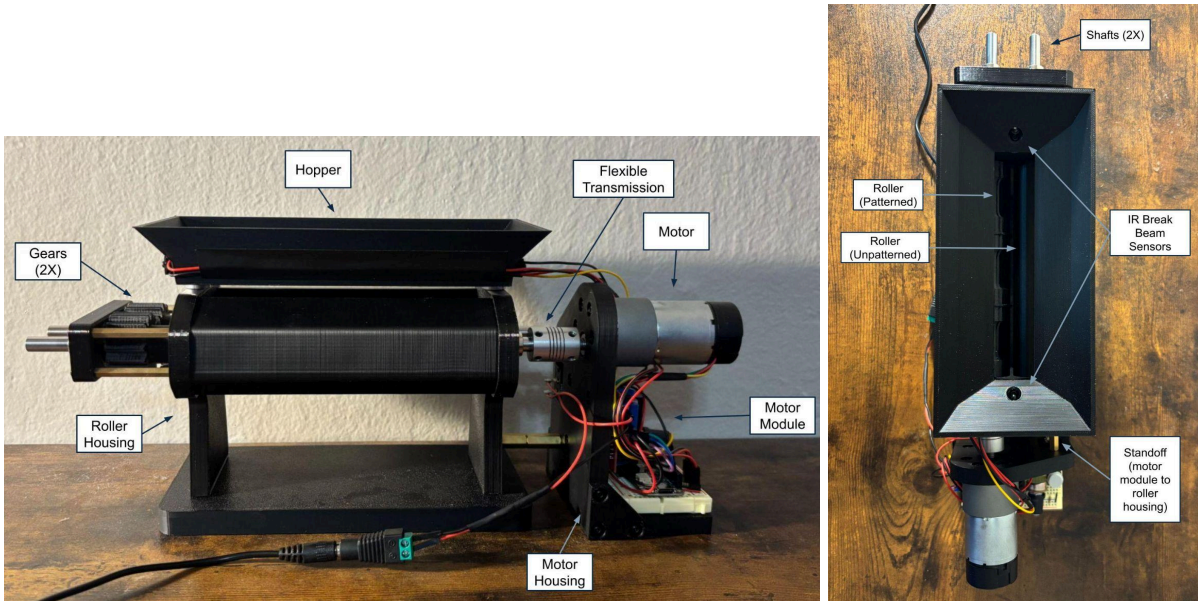
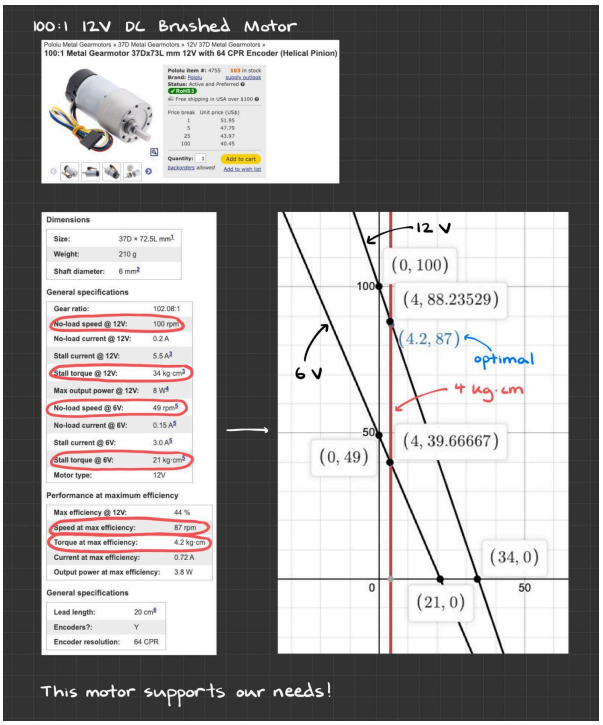


Figure 1: Side View of Ravioli Maker (Left) and Top View of Ravioli Maker (Right)

Calculations



Motor Selection Calculations

Roller Diameter: $25\ mm \rightarrow$

Roller Radius: $12.5\ mm$

Pasta Manual Roller Force: $2\ Kg$

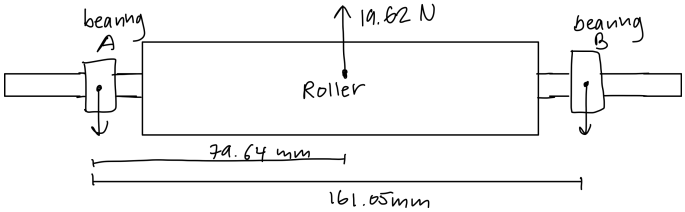
$$F = mg = 2 \cdot 9.81 = 19.62\ N$$
$$\tau = Fr = 19.62 \cdot 0.0125$$
$$= 0.24525\ Nm \approx 2.5\ Kg \cdot cm$$

Factoring in that we are pressing together 2 sheets of pasta and need to also cut the dough, we will set $3.5\ Kg \cdot cm$ as the target torque. As we are making a ravioli machine which is similar to a pasta maker, the rollers should rotate slowly. 1 rotation per second is reasonable. Therefore the target speed is 60 RPM.

Figure 2: Motor Selection Data

Bearing Selection Calculations

Using the previously calculated value for the force from the manual roller, we are able to calculate the radial force on the bearings. The rollers and gears are 3D printed and thus extremely light, and therefore have been ignored for this calculation. The largest force on the shaft comes from the resistance of cutting the dough, which acts perpendicular to the shaft. The shaft is symmetric, with both bearings equal distance from the center of the shaft, where the force is applied. Thus, both bearings take on the same amount of radial force.

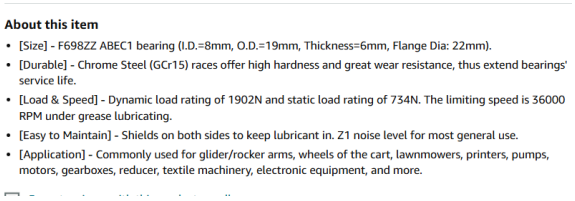


$$Bearing\ Load\ Factor\text{:}\ F_0 = 1.2$$

Pasta Manual Roller Force: $2\ Kg$

$$F = mg = 2 \cdot 9.81 = 19.62\ N$$
$$Bearing\ Force = 19.62N / 2 = 9.81\ N$$
$$Dynamic\ Load = F_0 \cdot Bearing\ Force = 11.772\ N$$

Our bearings must have a dynamic load of minimum $11.772\ N$, the ones we chose have a dynamic load of $1902\ N$, thus fitting our requirements.



Circuit Diagram

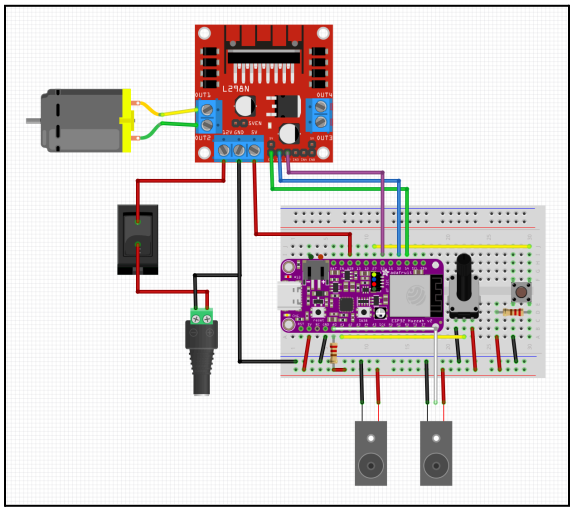


Figure 3. Ravioli Machine circuit diagram made with Fritzing

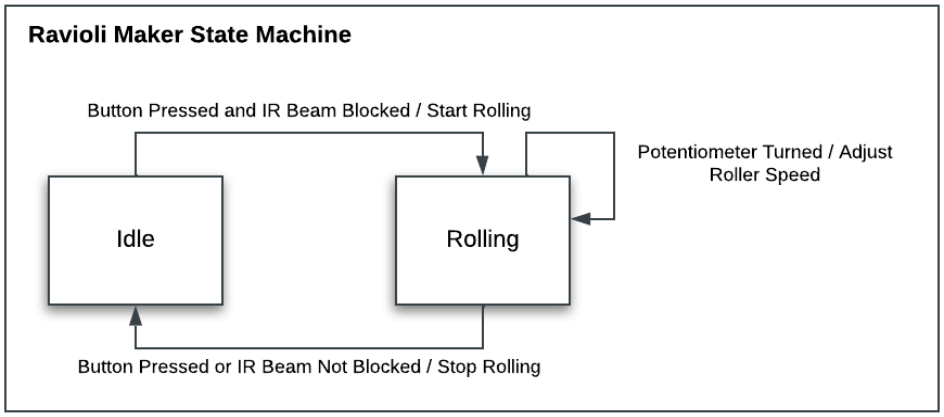


Figure 4. State Transition Diagram for our Ravioli Machine

Reflection

Our group did well coming up with ideas and improving upon previous designs by performing many tests with different parts. We were able to test the strength and durability of the hardware of our project before even finishing the code for the motor because we included the option to turn the machine via a hand crank, which allowed us to improve the design for the rollers early on in the process. We were also able to delegate tasks well depending on the skillsets of each member; one person did CAD and design work, one took the lead on coding, and the last helped with miscellaneous tasks and all manufacturing requiring the use of the Etcheverry Machine Shop.

Something we struggled with was following a schedule and meeting deadlines; we never established a formal timeline and self imposed deadlines to make sure all our components would be finished in time, which led to last minute rushing. When schedules became busy, we also had some struggles with communication. If we could redo anything, we would definitely establish a self imposed timeline at the start of the semester, and make agendas for our weekly meetings.

Appendix A

Bill of Materials

Required Parts	Listing Name	Quantity	Cost	Source Links
Motor	100:1 Metal Gearmotor 37Dx73L mm 12V with 64 CPR Encoder (Helical Pinion)	1	$51.95	https://www.pololu.com/product/4755
Microcontroller	Adafruit ESP32 Feather V2 - 8MB Flash + 2 MB PSRAM - STEMMA QT	1	$0.00	https://www.adafruit.com/product/5400
Motor Driver	L298N	1	$0.00	Amazon Link
Potentiometer	Potentiometer	1	$0.00	Spare
Button	Button	1	$0.00	Spare
IR Beam Sensor	IR Break Beam Sensor with Premium Wire Header Ends - 5mm LEDs	1	$5.95	https://www.adafruit.com/product/2168
Switch	RA1113112R	1	$0.68	https://www.digikey.com/en/products/detail/e-switch/RA1113112R/3778055
12V Adapter	12V Adapter	1	$11.99	Amazon Link
Barrel Jack Adapter	Barrel Jack Adapter	1	$0.00	Lab Kit
Breadboard	Breadboard	1	$0.00	Lab Kit
Shaft	8mm Shaft	2	$0.00	Spare
Shim	8mm Shim	6	$0.00	Spare
Belleville Washer	8mm Belleville Washer	2	$0.00	Spare

Shaft Collar	8mm Shaft Collar	4	$7.99	Amazon Link
Shaft Coupler	6mm to 8mm Shaft Coupler	1	$9.99	Amazon Link
Bearing	8mm Bearing	6	$11.99	Amazon Link
M2 Heat Set Insert	M2 Heat Set Insert	12	$18.99	Amazon Link
M3 Heat Set Insert	M3 Heat Set Insert	15	$0.00	Amazon Link
M4 Heat Set Insert	M4 Heat Set Insert	17	$0.00	Amazon Link
M6 Heat Set Insert	M6 Heat Set Insert	1	$0.00	Amazon Link
M3 Nut	M3 Nut	9	$0.00	Spare
M2 Screw	M2 Screw	10	$0.00	Spare
M3 Screw	M3 Screw	18	$0.00	Spare
M4 Screw	M4 Screw	13	$0.00	Spare
M6 Screw	M6 Screw	1	$0.00	Spare
M2 Washer	M2 Washer	2	$0.00	Spare
M3 Washer	M3 Washer	16	$0.00	Spare
M4 Washer	M4 Washer	1	$0.00	Spare
M6 Washer	M6 Washer	1	$0.00	Spare
M3 Standoff	M3 Standoff	4	$5.99	Amazon Link
M4 Standoff	M4 Standoff	1	$5.99	Amazon Link

M6 Standoff	M6 Standoff	2	$7.79	Amazon Link
PLA Filament	OVERTURE PLA Filament	1	$13.99	Amazon Link
PETG Filament	OVERTURE PETG Filament	1	$14.99	Amazon Link

Appendix B

CAD

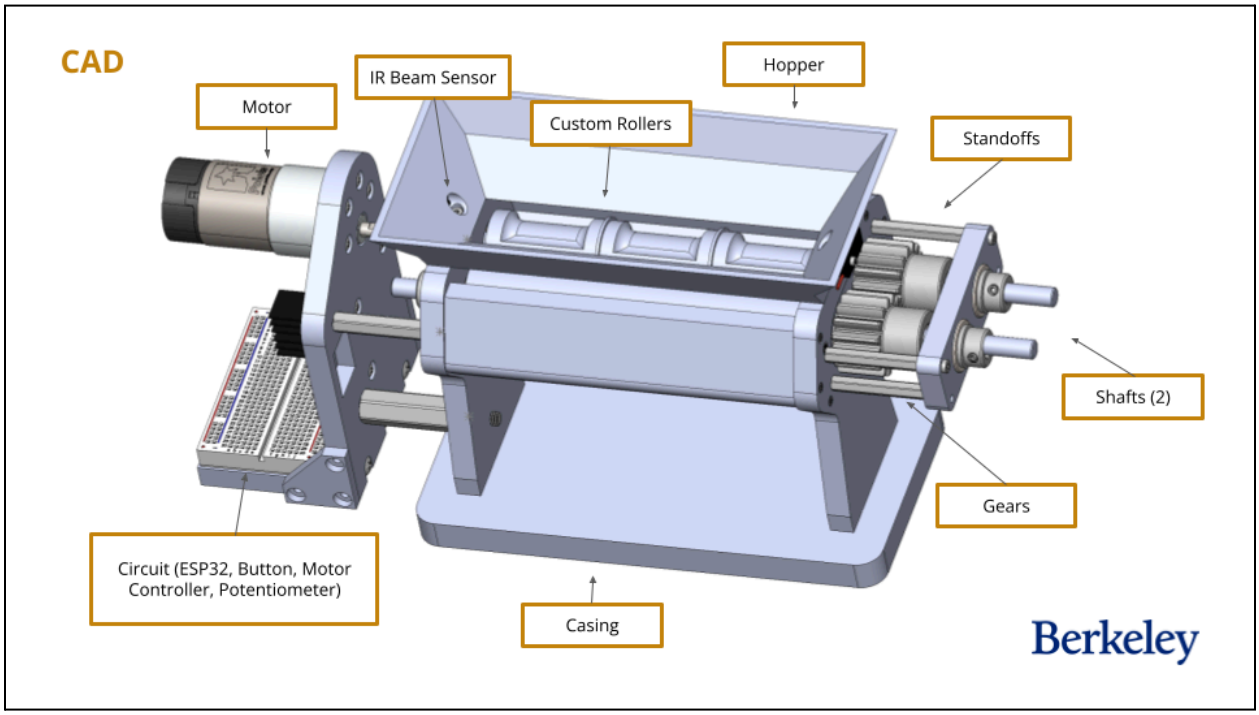


Figure #. Ravioli Machine Full Assembly

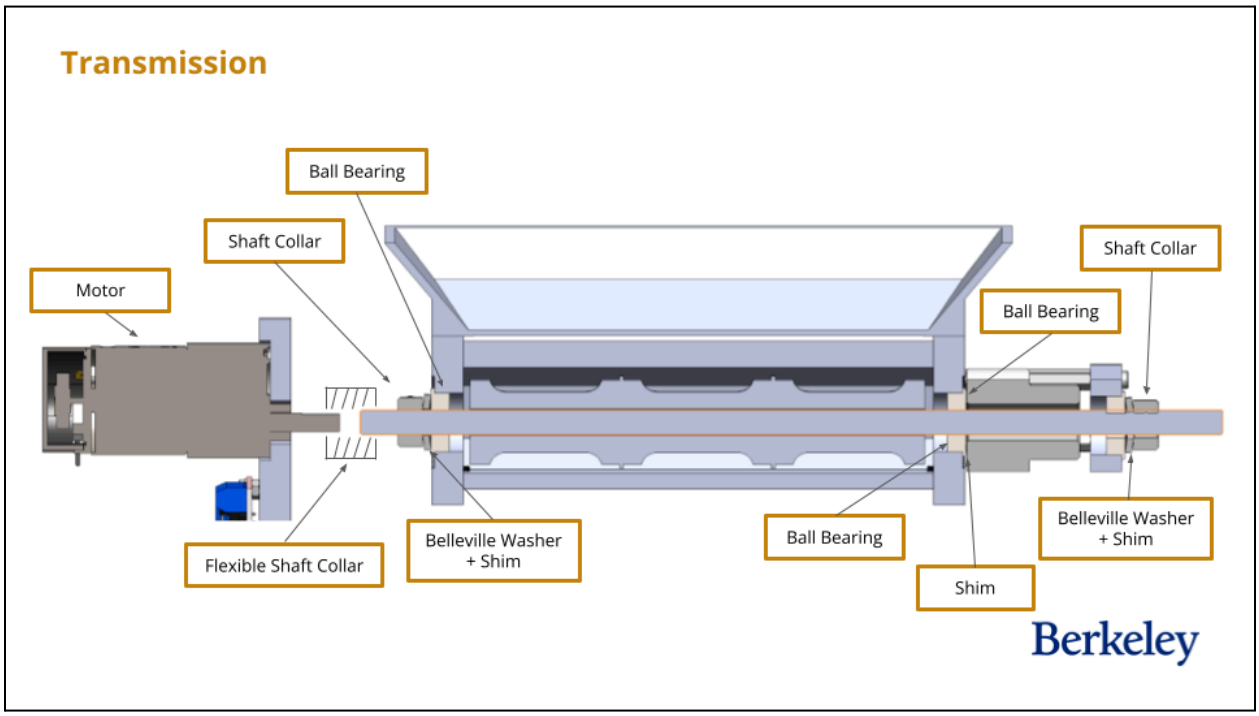


Figure #. Ravioli Machine Transmission

Appendix C

Code

```
#include <Arduino.h>

// Pins
#define button 27
#define pot 25
#define ir 26
#define bin1 32
#define bin2 33
#define pwm 14

// State
byte state = 0;

// PWM
const int pwmFreq = 5000;
const int pwmResolution = 8;

// Variables
int currPotValue = 0;
int prevPotValue = 0;
int difference = 0;
int motorSpeed = 0;

// Flags
volatile bool DEBOUNCINGflag = false;
volatile bool BUTTONflag = false;
volatile bool IRflag = false;

hw_timer_t *timer0 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;

// Button Interrupt
void IRAM_ATTR BUTTONisr() {
  BUTTONflag = true;
}
```

```
// IR Interrupt
void IRAM_ATTR IRisr() {
  if (digitalRead(ir) == LOW) {
    IRflag = true;
  } else {
    IRflag = false;
  }
}

// Timer Interrupt
void IRAM_ATTR onTime0() {
  portENTER_CRITICAL_ISR(&timerMux0);
  DEBOUNCINGflag = false;
  portEXIT_CRITICAL_ISR(&timerMux0);
  timerStop(timer0);
  BUTTONflag = false;
}

void setup() {
  pinMode(button, INPUT_PULLUP);
  pinMode(pot, INPUT);
  pinMode(ir, INPUT);
  pinMode(bin1, OUTPUT);
  pinMode(bin2, OUTPUT);

  attachInterrupt(button, BUTTONisr, RISING);
  attachInterrupt(ir, IRisr, CHANGE);

  ledcAttach(pwm, pwmFreq, pwmResolution);

  // Serial.begin(115200);

  timer0 = timerBegin(50000);
  timerAttachInterrupt(timer0, &onTime0);
  timerAlarm(timer0, 10000, true, 0);
  timerStop(timer0);
}
```

```
void loop() {

 delay(20);

 currPotValue = analogRead(pot);
 difference = abs((currPotValue - prevPotValue));
 prevPotValue = currPotValue;

 switch (state) {
  case 0: // Idle
   if (ButtonEvent() && IREvent()) {
    MotorOnService();
   }
   break;

  case 1: // Rolling
   if (ButtonEvent() || !IREvent()) {
    MotorOffService();
   } else if (PotEvent()) {
    MotorSpeedService();
   }
   break;

  default: // Error
   // Serial.println("SM ERROR");
   break;
 }
}

// Event Checker: Button Press
bool ButtonEvent() {
 if (BUTTONflag && !DEBOUNCINGflag) {
  portENTER_CRITICAL_ISR(&timerMux0);
  DEBOUNCINGflag = true;
  portEXIT_CRITICAL_ISR(&timerMux0);
  timerStart(timer0);
  return true;
 } else {
```

```
    return false;
  }
}

// Event Checker: Potentiometer Turn
bool PotEvent() {
  if (difference > 20) {
    return true;
  } else {
    return false;
  }
}

// Event Checker: IR Tripped
bool IREvent() {
  if (IRflag) {
    return true;
  } else {
    return false;
  }
}

// Service: Turn Motor On
void MotorOnService() {
  state = 1;
  motorSpeed = map(currPotValue, 0, 4095, 185, 230);
  digitalWrite(bin1, LOW);
  digitalWrite(bin2, HIGH);
  ledcWrite(pwm, motorSpeed);
  // Serial.println("State 1: Rolling");
}

// Service: Turn Motor Off
void MotorOffService() {
  state = 0;
  digitalWrite(bin1, LOW);
  digitalWrite(bin2, LOW);
  ledcWrite(pwm, 0);
```

```
  // Serial.println("State 0: Idle");
}

// Service: Adjust Motor Speed
void MotorSpeedService() {
  motorSpeed = map(currPotValue, 0, 4095, 180, 230);
  digitalWrite(bin1, LOW);
  digitalWrite(bin2, HIGH);
  ledcWrite(pwm, motorSpeed);
  // Serial.print("State 1: Set Motor Speed to ");
  // Serial.println(motorSpeed);
}
```