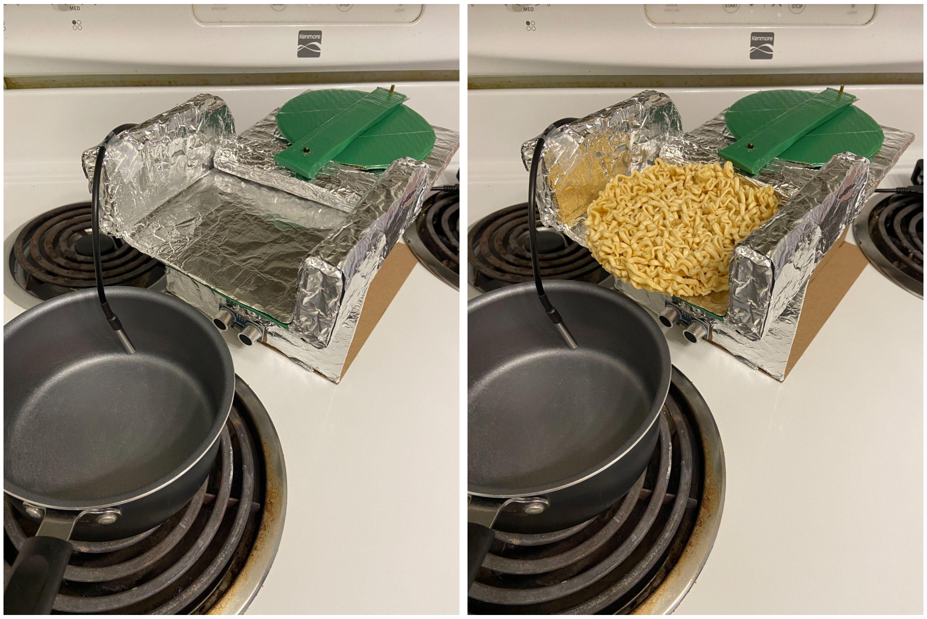
MAE 3780 Janis Song
November 3, 2021 jjs468

Individual Robot Final Project Report - RoboRamen©

I. Overview

Ramen is cheap and easy, but can still be time-consuming for a busy college student. That is why I am creating RoboRamen© - a robot that can make ramen for you! All the person needs to do is get a pot of water and put the soup powder in, place the ramen noodles on the mount of the robot, and then turn on the heat. After that, the robot takes care of the rest. A temperature sensor is placed in the pot of water to determine when it reaches boiling point (about 100℃, this can be adjusted in the Arduino code to account for different altitudes). Once that temperature is reached, it will trigger a slider-crank mechanism connected to the continuous servo motor to push the ramen noodles into the pot of water. That will also trigger a timer to be set (which can also be adjusted in the code), and when the noodles are done, a tune will play from the piezo buzzer. The robot also features an ultrasonic sensor that will also trigger the piezo buzzer to beep if the robot is placed too close to the hot pot (I set a safe distance of 4 cm).



Fully Assembled Design (without and with ramen noodles)

II. Design Considerations

For someone attempting to replicate this project, I would suggest they 3D print the slider-crank mechanism. Because of the $20 budget constraint, I was very hesitant to utilize the 3D printing lab, since I assumed it would be very expensive to print the mechanism required for my project and I already had to purchase additional materials for my project. However, it proved to be more time-consuming to stabilize the cardboard and manufacture the crank-mechanism by hand, and a lot less precise, which is essential when operating a slider-crank. As it turns out, it would have been within budget to utilize the 3D printer, and a lot more efficient and stable, which is what I would do if I could do this project over again. I would also have incorporated an extra servo motor in my design to reel in the temperature sensor in case it gets heat damage from the boiling water. This would definitely be of importance if the robot undergoes repeated usage.

If I had more time and money, I would 3D print the mount and the slider crank mechanism and overlay it with a spray that can protect it from melting from the heat of the boiling water and hot pot. I would also improve the overall aesthetics of the robot design itself, incorporating the more aesthetically pleasing design I had originally made in CAD, which proved to be difficult to replicate when utilizing flimsy cardboard and a lot of tape. I would focus on increasing the stability and longevity of the device, and spend more time on the overall look and feel of the robot itself.

III. Assembly Instructions

Building the Circuit for the RoboRamen©:

1. Build the circuit according to the circuit diagram (***refer to Appendix B)***.
    a. Test the circuit with the Arduino code to ensure it is working properly before moving on to the next steps. Tape down or solder any wires that are loose. Add a 3-wire extension to the prongs of the ultrasonic sensor so it can extend to the front of the robot.

Building the Base Mount for the RoboRamen©:

2. Cut out the side base walls 1 & 2, front wall, and bottom base plate out of cardboard according to the dimensions in the appendix (***refer to Figures 1-4)***.

3. Construct the base of the RoboRamen© by taking the side base walls and front base wall ***(refer to Figures 1-3)*** and super-gluing or hot gluing them to the bottom base plate ***(refer to Figure 4)***.
    a. ***Refer to Figure 5-6*** for what the assembly should look like and specific measurements.

4. To the front wall of the base, tape down the ultrasonic sensor so it looks like ***Figure 7***. Add foil to cover up the wires and secure the ends of it with tape ***(refer to Figure 8)***.
    a. The wires of the ultrasonic sensor should go through the hole in the base front wall ***(refer to Figure 3)***.

5. Place the circuit and Arduino into the base mount with the A to B USB Cable going through the hole in the *base side wall (2)* ***(refer to Figure 2)***. Place the

continuous servo motor and the temperature sensor outside of the base mount; we will incorporate them later into the design.

   a. ***Refer to Figure 9*** (the yellow highlighted circle) for what this should look like.

Building the Slider-Crank Mechanism for the RoboRamen©:

6. Cut out 10 pieces of cardboard to the dimensions of ***Figure 10*** (the arc in the piece is purely aesthetic and can be done without measurements). Layer five of the pieces together and bound securely with duct tape, all around the piece until it is covered by duct tape. Repeat with the other 5 pieces. These are the side walls of the slider crank, which are reinforced by the multiple layers of cardboard.

7. Cut out 3 pieces of cardboard to the dimensions of ***Figure 11***. Layer the pieces together and bound securely with duct tape, all around the piece until it is covered by duct tape. This is your slider crank pusher. Poke a hole only halfway through the part as indicated in the drawing, big enough for the standoff to fit.
   a. Overlay this part with foil so it has a smoother contact point with the walls of the slider crank.

8. Cut out 2 pieces of cardboard to the dimensions of ***Figure 12***. Layer the pieces together and bound securely with duct tape, all around the piece until it is covered by duct tape. This is your slider crank arm.
   a. Poke holes near the top and bottom of the crank arm (as indicated in ***Figure 12***) so the standoff can fit through. Make sure it isn’t too close to the edges.

9. Finally, cut a single piece of cardboard to the dimensions of ***Figures 13-14***. These are your slider crank base and wheel. Put a layer of duct tape over each piece for increased stability. Poke a hole through the slider crank wheel as shown in ***Figure 14*** so that a standoff can fit through.

10. Tape the continuous servo motor down to the bottom of the slider crank base as shown in ***Figure 15***.

11. Place and duct tape the slider crank walls down onto the slider crank base as shown in ***Figure 16.*** Make sure the slider crank pusher can move smoothly between them.
    a. Overlay the whole structure with foil (to make it a smoother surface so the crank can operate easily) and secure it all down. Make sure to keep a small opening for the servo motor to peek through.

12. Tape the servo motor attachment to the bottom of the slider crank wheel ***(refer to Figure 17)***.

13. Attach the slider crank arm to the edge of the wheel using the pre-poked holes and a standoff. Secure the male end of the standoff with a hex nut to ensure it stays together. Secure the other end of the slider crank arm to the slider crank pusher using the pre-poked holes and the standoff. Have the female end of the standoff go first so you can secure the male end (that sticks out) with a hex nut.
    a. ***Refer to Figure 18*** for how the final assembly should look.

14. Attach the servo motor attachment to the servo motor on the slider crank base, and adjust the slider crank pusher so it is aligned between the slider crank walls ***(refer to Figure 19)***.

Putting the RoboRamen© all together:

15. Place the slider crank mechanism on top of the base mount of the RoboRamen©. Tape the temperature sensor to the side of the slider crank wall, angling it specifically so it would be submerged in the water for your pot. ***Refer to Figures 20-21*** for how the finished assembly should look!

IV. Operation Instructions

1. Place the RoboRamen© on the stovetop, next to a pot filled with water.

2. Use the A to B USB Cable to connect the Arduino board to the computer.

3. Edit the RoboRamen© code to the boiling point temperature of your altitude, and the buzzer timer to the cook time of your ramen. Upload the code.

4. Adjust the position of the robot if the buzzer starts beeping (the hot pot will be too close to the robot otherwise!)

5. Place the noodles on top of the mount, add your soup base and flakes into the pot, and place the temperature sensor into the pot.

6. Let the robot do its thing, walk away, and get some work done!

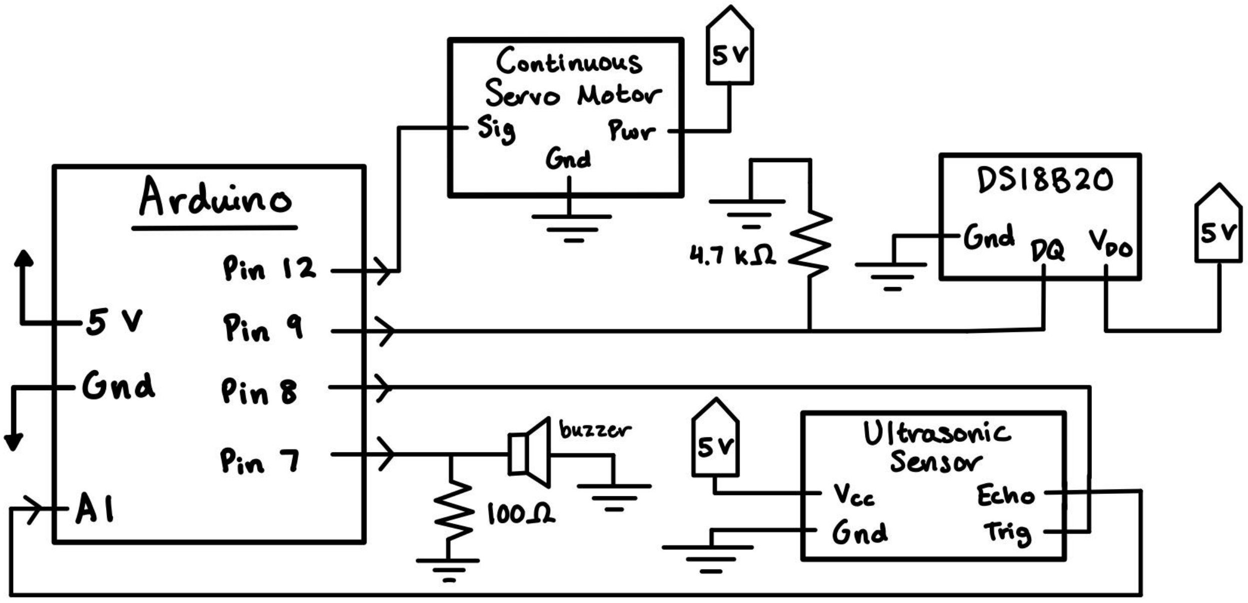
7. The buzzer will play a tune and your ramen is all ready. Enjoy!

Appendices

I. Appendix A: Bill of Materials (BoM)

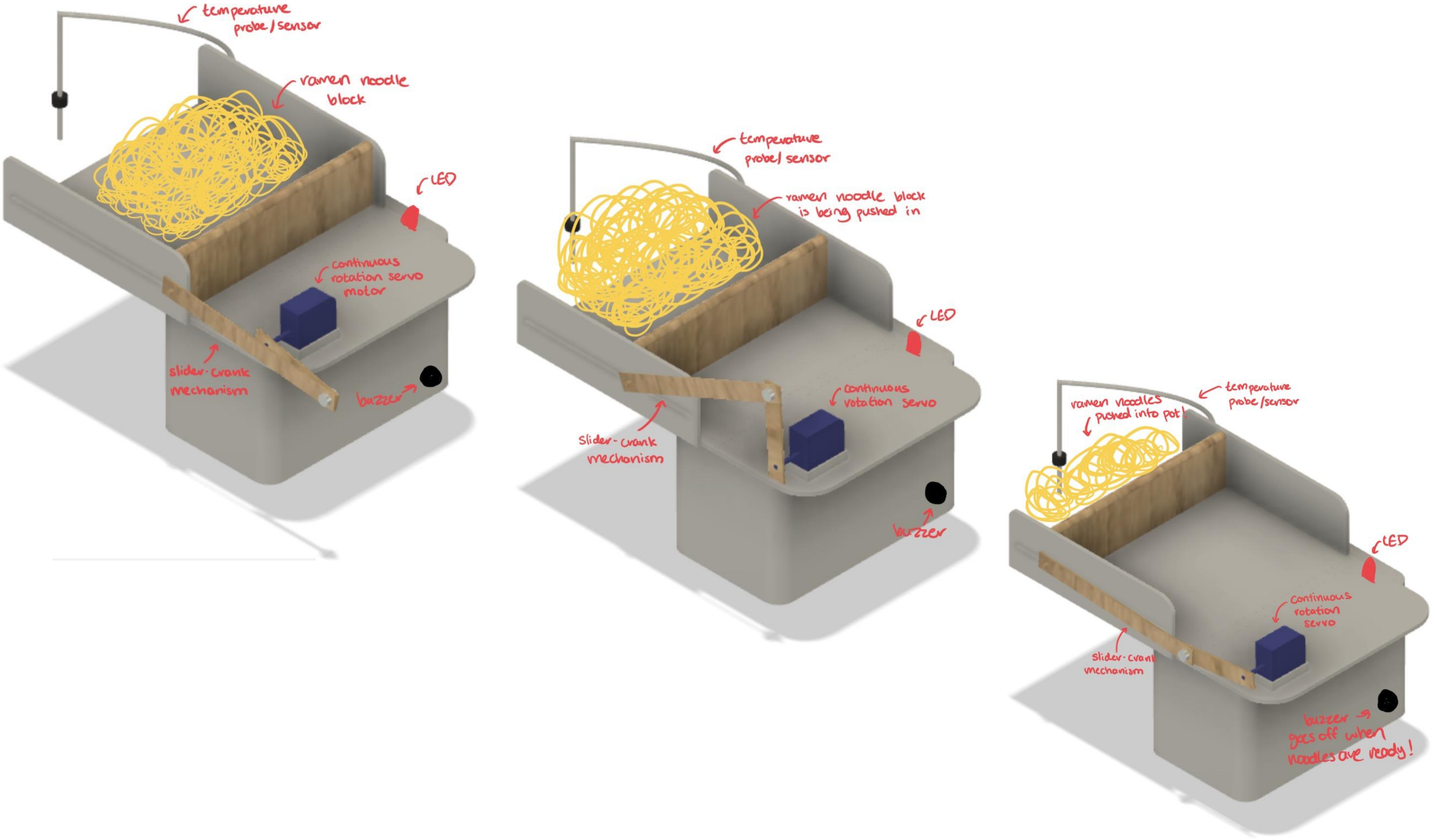
Item	Part Name	Description	Vendor/ Source	Part #	Quantity	Price (Each/ Per Unit)	Sub Total
1	Arduino Board	Standard Arduino board	Digi-Key	1050-1024-ND	1	$20.90	$20.90
2	USB Cable	USB Cable A to B	Monoprice	39918	1	$1.09	$1.09
3	Breadboard	Standard Breadboard	Newark	79X3922	1	$2.71	$2.71
4	Servo Motor	Continuous micro servo motor	DFRobot	SER-0043	1	$3.90	$3.90
5	Electrical Wire	Assorted lengths and colors, Wire Kit	Amazon: Austor	B07PQKNQ22	1	$2.17	$2.17
6	3-Wire Extension	Wire extension with pin holes	Digi-Key	1568-1930-ND	2	$1.35	$1.35
7	Resistor	4.7k ohm	Digi-Key	4.7kQBK-ND	1	$0.01	$0.01
8	Resistor	100 ohm	Digi-Key	100QBK-ND	1	$0.01	$0.01
9	Ultrasonic Sensor	Ultrasonic distance sensor	Amazon	B07RGB4W8V	1	$1.66	$1.66
10	Temperature Sensor	Waterproof	Amazon	TSDS18B20-1M	1	$3.80	$3.80
11	Buzzer	Piezo buzzer (3V)	Digi-Key	445-2525-1-ND	1	$0.61	$0.61
12	Cardboard	8.5" x 11" 22Pt Cardstock	Lab Section		7	$0.15	$1.05
13	Aluminum Foil	Standard kitchen foil	Scavenged		1	$0.50	$0.50
14	Duct Tape	Duct tape	Scavenged		1	$0.50	$0.50
15	Scotch Tape	Scotch tape	Scavenged		1	$0.01	$0.01
16	Hex Nut	M2 hex nut	Scavenged		2	$0.02	$0.04
17	Standoff	M2 brass, M-F, cylindrical	Scavenged		2	$0.01	$0.02
				Total Including Kit Items:			$40.33
				Total Without Kit Items:			$8.19

II. Appendix B: Circuit Diagram



III. Appendix C: CAD files and drawings

Initial CAD Design (with initial slider-crank design):


temperature probe/sensor
ramen noodle block
LED
continuous rotation servo motor
slider-crank mechanism
buzzer
temperature probe/ sensor
ramen noodle block is being pushed in
LED
continuous rotation servo
slider-crank mechanism
buzzer
ramen noodles pushed into pot!
temperature probe/sensor
LED
continuous rotation servo
slider-crank mechanism
buzzer → goes off when noodles are ready!

Assembly Instruction Supplemental Images:

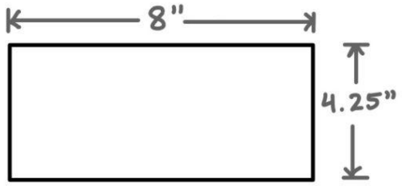


Figure 1: Cardboard Base Side Wall (1)

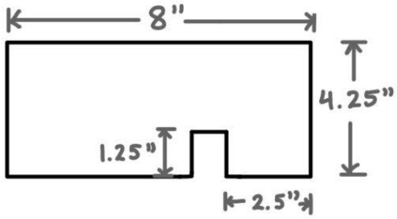


Figure 2: Cardboard Base Side Wall (2)

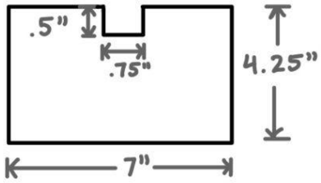


Figure 3: Cardboard Base Front Wall

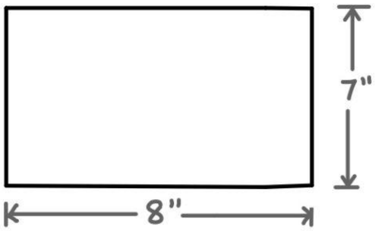


Figure 4: Cardboard Base Bottom Plate

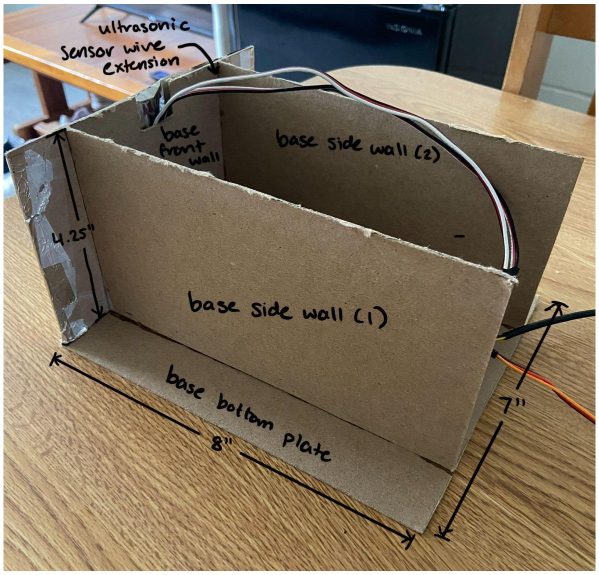


Figure 5: Assembly of base of robot

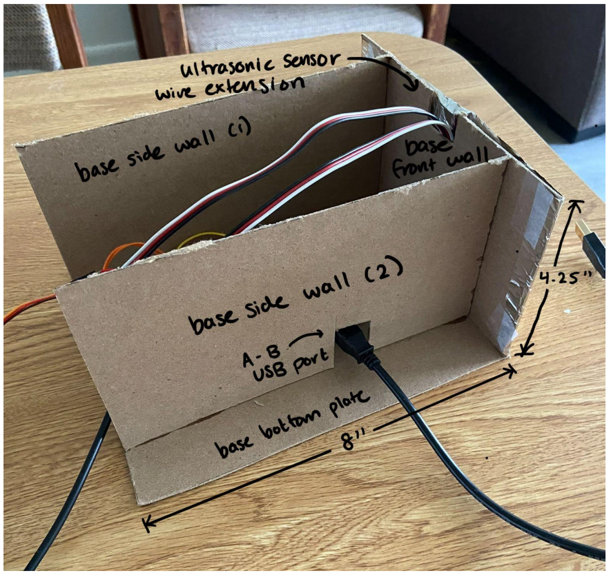


Figure 6: Other side of Assembly of robot

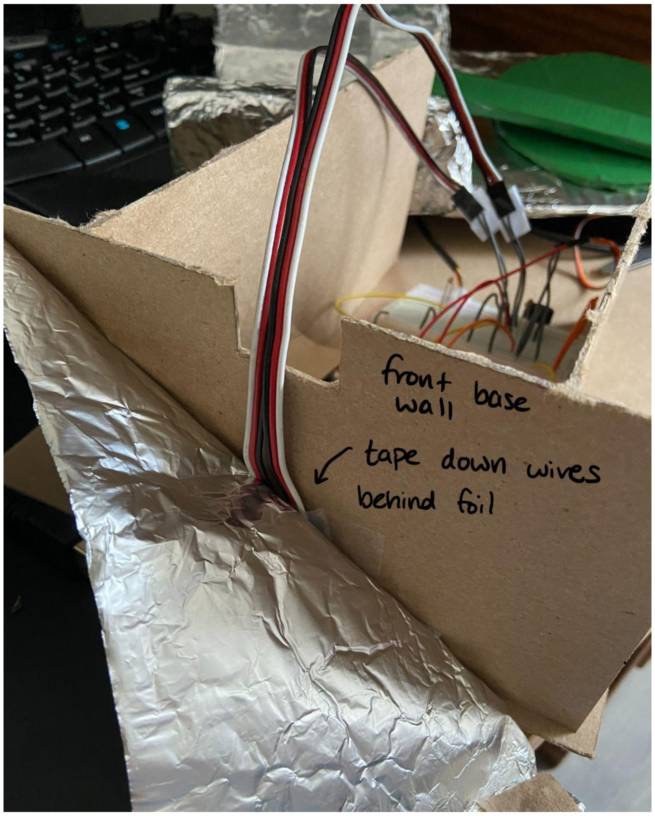


Figure 7: Ultrasonic Sensor Wire Taped Down

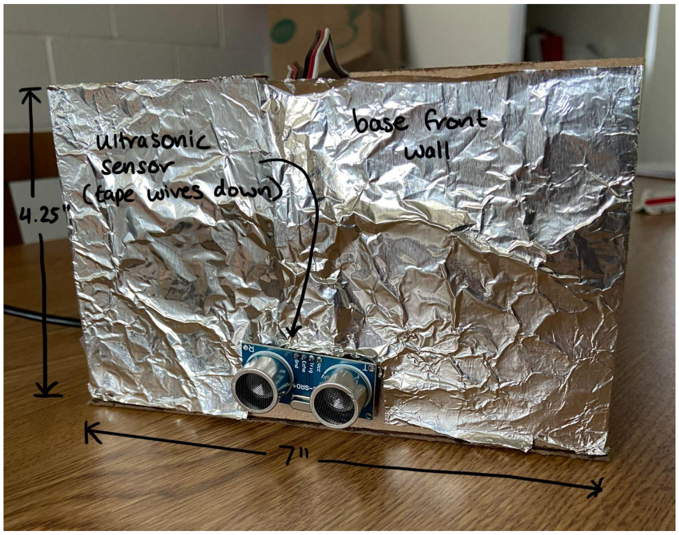


Figure 8: Front Wall Assembly w/ Ultrasonic Sensor

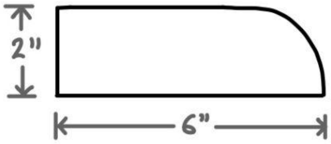


Figure 10: Cardboard Slider Crank Wall

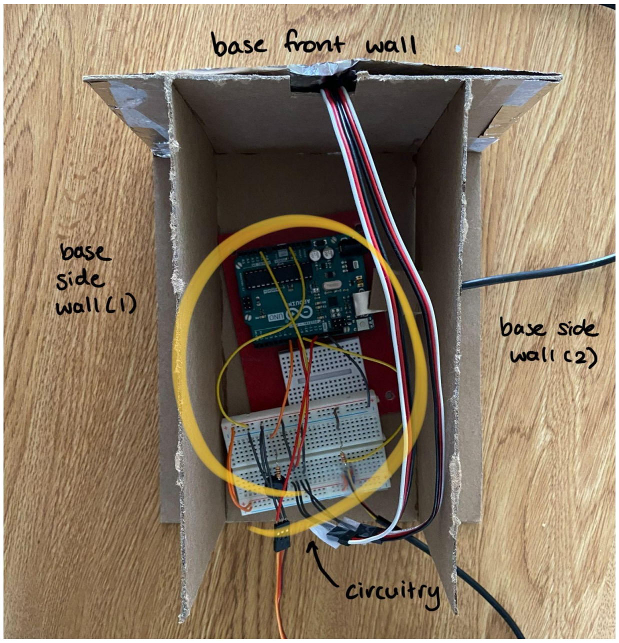


Figure 9: Circuitry with Finished Base Mount

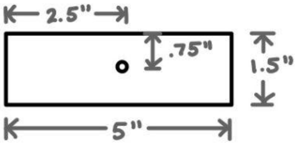


Figure 11: Cardboard Slider Crank Pusher

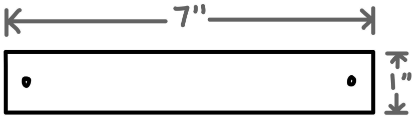


Figure 12: Cardboard Slider Crank Arm

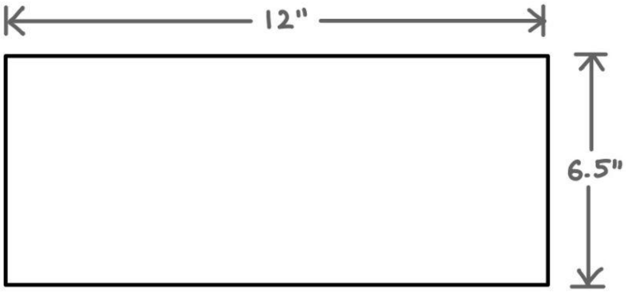


Figure 13: Cardboard Slider Crank Base

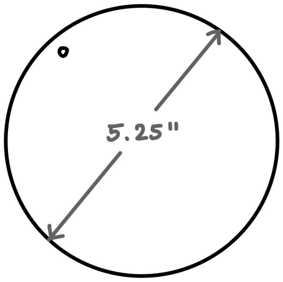


Figure 14: Cardboard Slider Crank Wheel

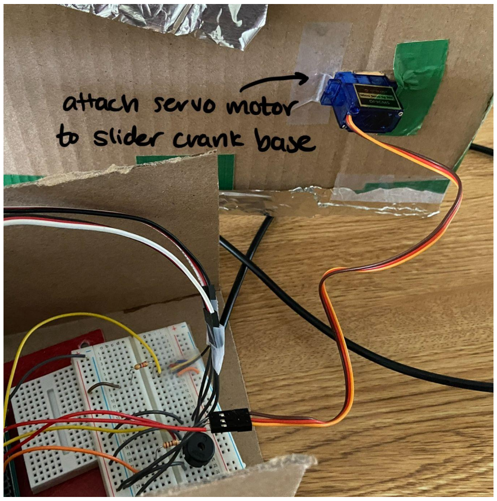


Figure 15: Servo Motor Attachment to Base

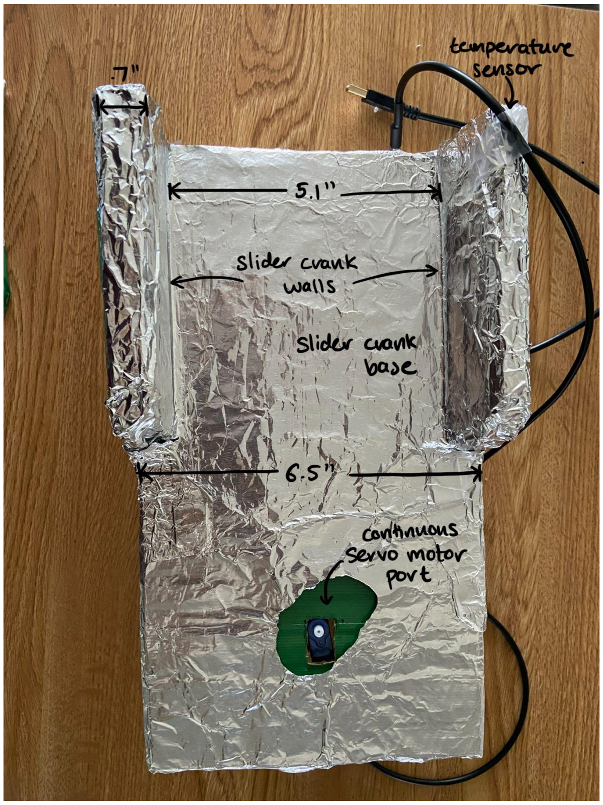


Figure 16: Slider Crank Wall Attachment, Foil Overlay

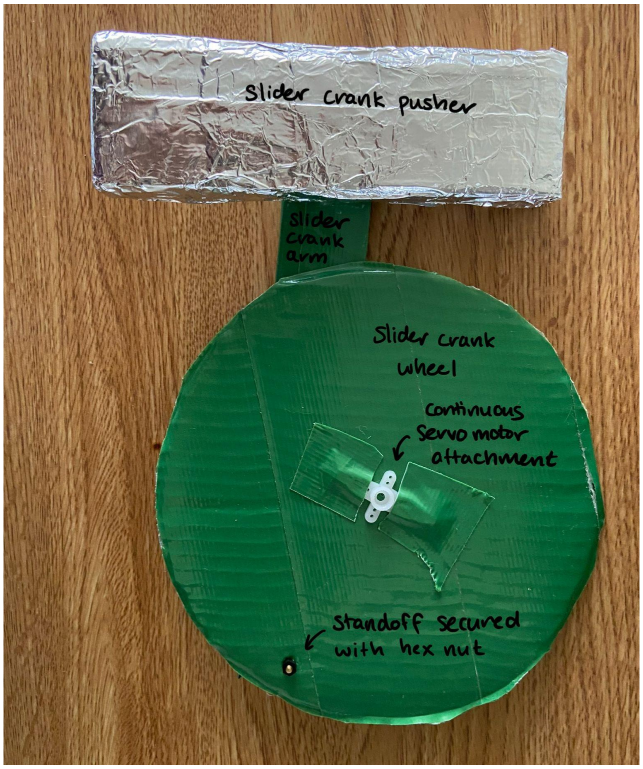

Slider crank pusher
Slider crank arm
Slider crank wheel
Continuous servo motor attachment
Standoff secured with hex nut

Figure 17: Bottom of Slider Crank Mechanism

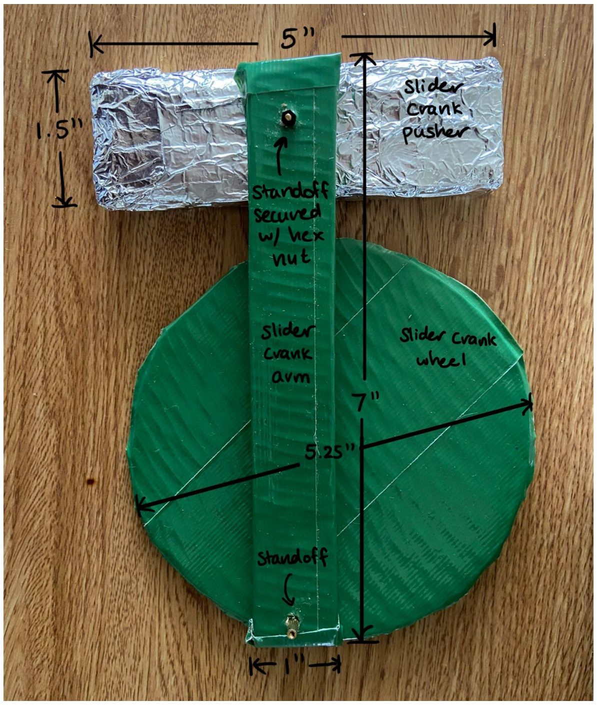


Figure 18: Top of Slider Crank Mechanism

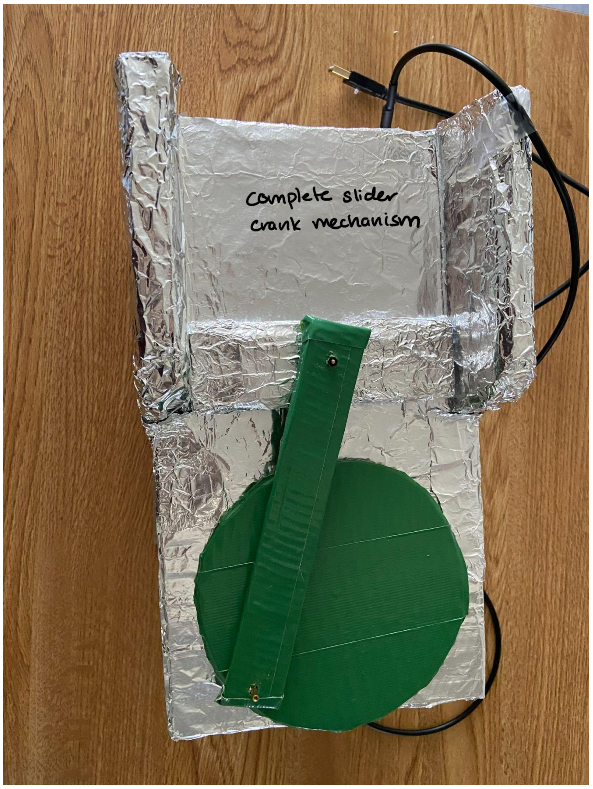


Figure 19: Finished Slider Crank Mechanism

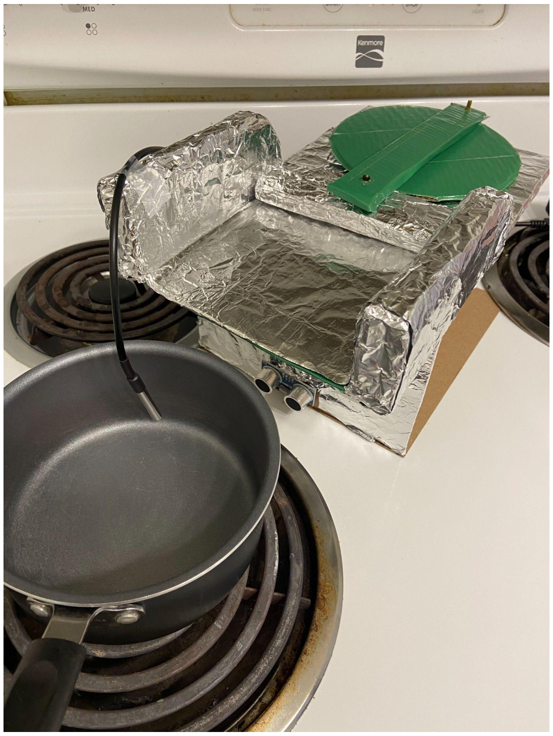


Figure 20: Finished RoboRamen©

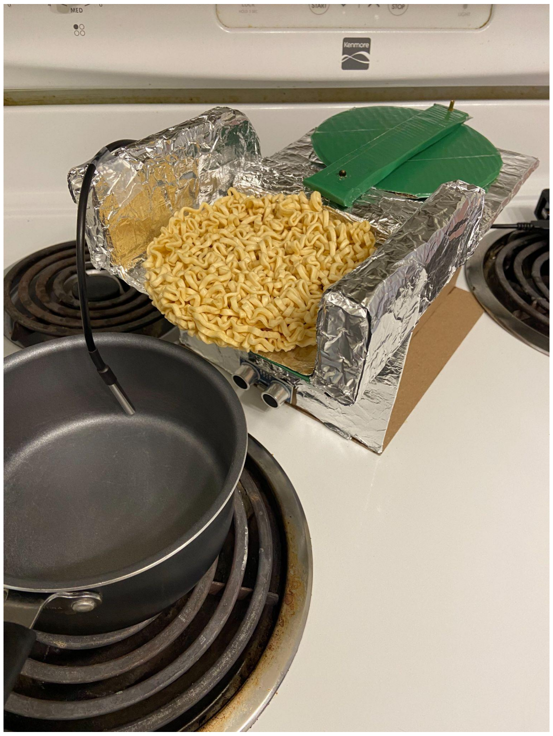


Figure 21: Finished RoboRamen© with Ramen

IV. Appendix D: Commented Arduino Code

```
// C++ code
//This code is for my RoboRamen. The ultrasonic sensor senses how far it is from the
//hot pot and causes the red LED light to turn on if it is closer than 4 cm to the hot pot.
//The temperature sensor, when it senses 100 degrees Celsius or the water boiling,
//causes the servo motor to turn (which operates a slider-crank mechanism) that pushes
//the ramen noodles into the pot. A timer is set from there with the delay function and
//after it is done, the alarm goes off to alert that the ramen is done. The loop for the code
//stops after that as well.

#include <Servo.h>
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
```

```
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
```

```
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
#define REST      0
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 9

// Setup a oneWire instance to communicate with any OneWire device (for temperature
sensor)
OneWire oneWire(ONE_WIRE_BUS);

// Pass oneWire reference to DallasTemperature library (for temperature sensor)
DallasTemperature sensors(&oneWire);

int cm = 0;     // variable to store measured distance (in cm) for ultrasonic sensor

// Code to get distance from the ultrasonic sensor
```

```
long readUltrasonicDistance(int triggerPin, int echoPin)
{
  pinMode(triggerPin, OUTPUT);  // Clear the trigger
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  // Sets the trigger pin to HIGH state for 10 microseconds
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  // Reads the echo pin, and returns the sound wave travel time in microseconds
  return pulseIn(echoPin, HIGH);
}

int celsius = 0; // variable to store celsius temperature from sensor

int pos = 0;    // variable to store the servo position

Servo myservo; // create servo object to control servo reeling in temperature probe and
pushing crank

int x = 1;      //variable to stop the loop when the ramen is done

// for buzzer melody:
int tempo = 114; // set tempo of buzzer song

// set melody for the buzzer alarm

int melody[] = {

  // Never Gonna Give You Up - Rick Astley
  // Arranged by Chlorondria
  NOTE_A4,16, NOTE_B4,16, NOTE_D5,16, NOTE_B4,16,
  NOTE_FS5,-8, NOTE_FS5,-8, NOTE_E5,-4, NOTE_A4,16, NOTE_B4,16,
NOTE_D5,16, NOTE_B4,16,

  NOTE_E5,-8, NOTE_E5,-8, NOTE_D5,-8, NOTE_CS5,16, NOTE_B4,-8,
NOTE_A4,16, NOTE_B4,16, NOTE_D5,16, NOTE_B4,16, //18
  NOTE_D5,4, NOTE_E5,8, NOTE_CS5,-8, NOTE_B4,16, NOTE_A4,8, NOTE_A4,8,
NOTE_A4,8,
  NOTE_E5,4, NOTE_D5,2, NOTE_A4,16, NOTE_B4,16, NOTE_D5,16, NOTE_B4,16,
  NOTE_FS5,-8, NOTE_FS5,-8, NOTE_E5,-4, NOTE_A4,16, NOTE_B4,16,
NOTE_D5,16, NOTE_B4,16,
  NOTE_A5,4, NOTE_CS5,8, NOTE_D5,-8, NOTE_CS5,16, NOTE_B4,8, NOTE_A4,16,
NOTE_B4,16, NOTE_D5,16, NOTE_B4,16,
  NOTE_D5,4, NOTE_E5,8, NOTE_CS5,-8, NOTE_B4,16, NOTE_A4,4, NOTE_A4,8,
//23
  NOTE_E5,4, NOTE_D5,2, REST,4,
};

// more buzzer melody semantics
int notes = sizeof(melody) / sizeof(melody[0]) / 2;

// this calculates the duration of a whole note in ms
int wholenote = (60000 * 4) / tempo;

int divider = 0, noteDuration = 0;

void setup()
{
  // attach led to pin 13
  pinMode(11, OUTPUT);

  sensors.begin();  // Start up the library for temperature sensor
  Serial.begin(9600);

  // initialize serial communication for buzzer and attaches it to pin 7
  Serial.begin(9600);
  pinMode(7, OUTPUT);

  // attaches the servo for reeling in the temp probe on pin 12 to the servo object
  myservo.attach(12); // attach servo to pin 12
}

void loop()
{
  if (x==1) {
    // Set distance variable (in cm) for ultrasonic sensor
    cm = 0.01723 * readUltrasonicDistance(8, A1);
    delay(10); // Delay a little bit to improve simulation performance
    Serial.print(cm);
    Serial.print("cm, ");

    // alarm if the object is too close:
    if(cm <= 4) {
      tone(7, 1000, 100);
      tone(7, 1000, 100);
      tone(7, 1000, 100);
    }

  // Send the command to get temperatures
  sensors.requestTemperatures();

  //print the temperature in Celsius
  Serial.print("Temperature: ");
  Serial.print(sensors.getTempCByIndex(0));
  Serial.print(" C  |  ");

    // if boiling point is reached, trigger the servo mount to rotate
    // and push the ramen noodles into the pot and reel the temp probe in
    if (sensors.getTempCByIndex(0) >= 25) {
      myservo.write(120);              // tell servo to rotate and push in ramen
      delay(2650);
      myservo.write(90);               // stop the servo rotation
      delay(270000);                   // set 4.5 min timer for the ramen noodles to cook
      for (int thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {

        // calculates the duration of each note
        divider = melody[thisNote + 1];
        if (divider > 0) {
            // regular note, just proceed
            noteDuration = (wholenote) / divider;
        } else if (divider < 0) {
            // dotted notes are represented with negative durations!!
            noteDuration = (wholenote) / abs(divider);
            noteDuration *= 1.5; // increases the duration in half for dotted notes
        }

        // we only play the note for 90% of the duration, leaving 10% as a pause
        tone(7, melody[thisNote], noteDuration * 0.9);

        // Wait for the specific duration before playing the next note.
        delay(noteDuration);

        // stop the waveform generation before the next note.
        noTone(7);
      }
      x = 0;            //cause if statement to be false so the loop stops running
}
  }
}
```