Feedback Coffee Machine

Group 11: Chan Kyu Lee, Ishaan Issar, Matthew Chen, Noah Park
Dept. of Mechanical Engineering, University of California at Berkeley
(Date: 05/01/2023)

1. Introduction

Feedback Coffee Machine is an AI/ML assisted coffee maker that offers several advanced features to enhance the brewing experience of the users. It is designed in clear acrylic to let users see how coffee is actually brewed, and well-designed to prevent water leakage problems from water tank or tubing, and heat transfer issues from heating elements. It can adjust the strength of the coffee in real-time according to the user's preferences. It also monitors the daily caffeine intake of the user, allowing them to keep track of their consumption. Furthermore, the coffee maker is designed to efficiently dispose of waste without users doing it with their own hand. And finally, Feedback Coffee is designed to connect you to your friends online and keep in touch through America's most consumed beverage; coffee!

2. About the Project

2.1 Project Description

2.1.1 Hardware Design

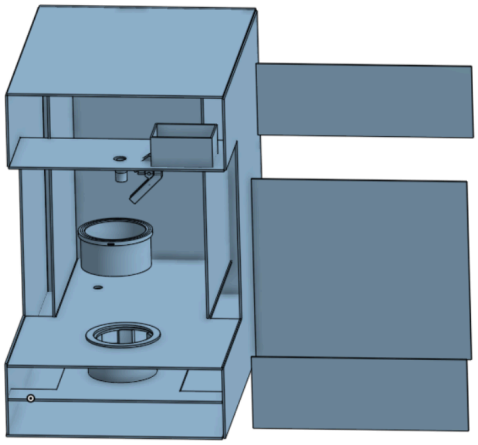


Figure 1. Initial CAD design

- Feedback Coffee Machine was designed to transfer water through the one way valve attached to the water tank and through the high temperature food-grade silicone tubing to reach the heating element. As the water is heated by the heating element, it turns into steam and builds up the high pressure. The pressure of the steam forces water up to reach the nozzle of the coffee machine. While water transfers through the tubing, the motor of the coffee grounds dispenser is generated to dispense the specific amount of the coffee grounds to the filter. Boiled water dispensed through the nozzle goes through the filter cup with the coffee grounds to dispense the coffee into the cup under the filter. After coffee brewing is done, the servo motor attached to the filter is generated to flip the filter 180 degrees and shake off the remaining coffee grounds.

- The entire housing of the coffee machine was made with laser cutted acrylic, with a laser etched wood plaque. Motor mounts, coffee filter, nozzle, and coffee ground storage was made out of 3D printing.

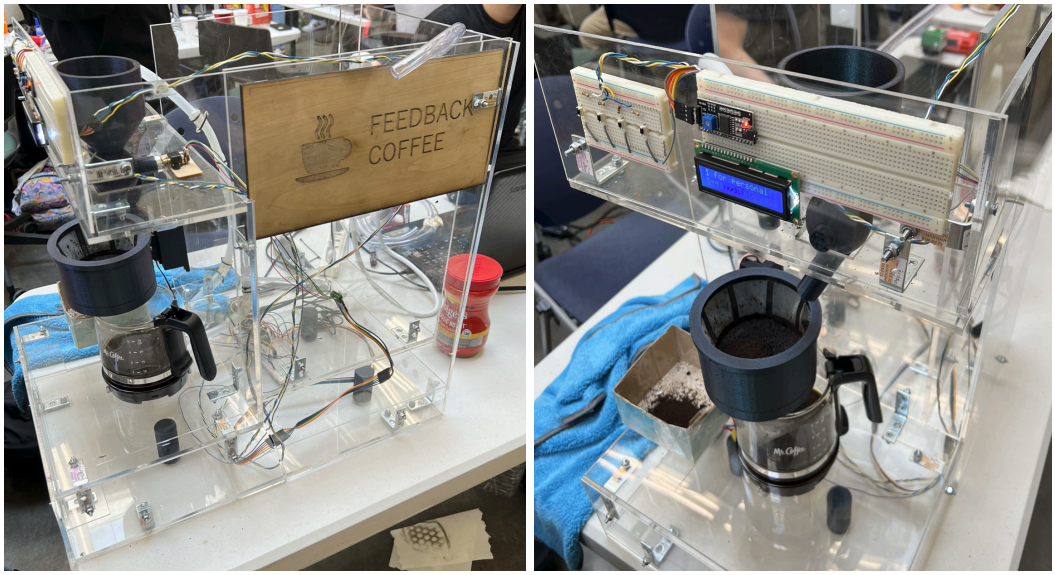


Figure 2. Feedback coffee machine

2.1.2 Circuit Design

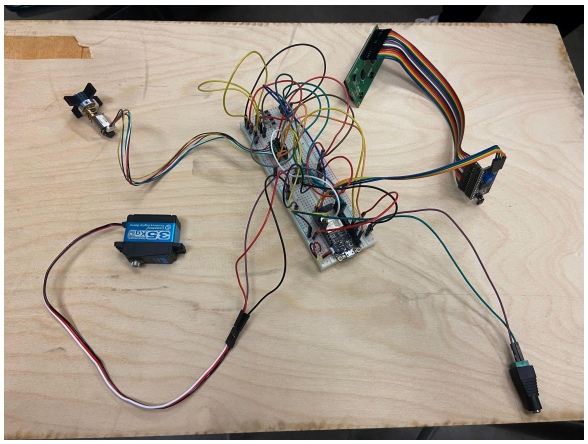


Figure 3. ESP32 circuit with 5V supply attached to it

- The circuit was designed to have 2 different voltage lines. The regular 120V and 5V lines. The 120V is used for the heating element and the 5V for the ESP32, servo motor, LCD, motor drive and motor.

2.1.3 Software (Code)

- For the actual code, it is attached on the appendix.
- First, we made different phases in the code to show different information on the LCD display. Each phase is entered/exited by pressing the buttons and once the required inputs, such as number of tablespoons for coffee grind or cups to brew are satisfied, it activates the coffee grind dispenser motor and waits until the coffee is brewed. Finally, once it is brewed, the user can press the button 4 twice to activate the servo motor and clean the filter. Also, the number of cups of coffee brewed and caffeine intake data are sent to a website where people can share and review their activities.

2.2 Innovative Components

2.1.1 Hardware

- Self Cleaning reusable filter cup : 3D printed filter cup is attached to servo motor and coded to flip 180 degree and shake off to clean the remaining grounds after coffee is made.

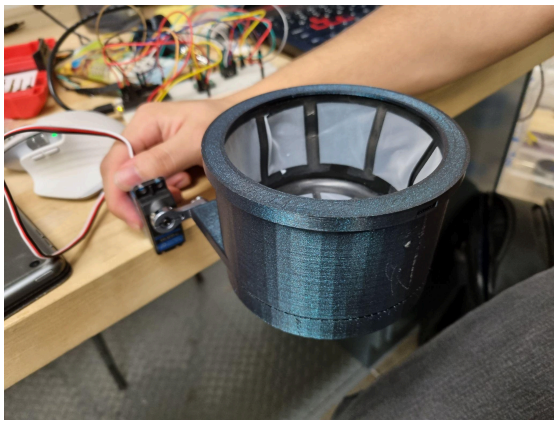


Figure 4. Filter cup and mounted servo motor

- Self Dispensing coffee grounds : 3D printed coffee ground storage is attached to the motor and 3D printed motor case. The rotation of the motor adjusts and dispenses the exact amount of coffee ground determined by user input data.

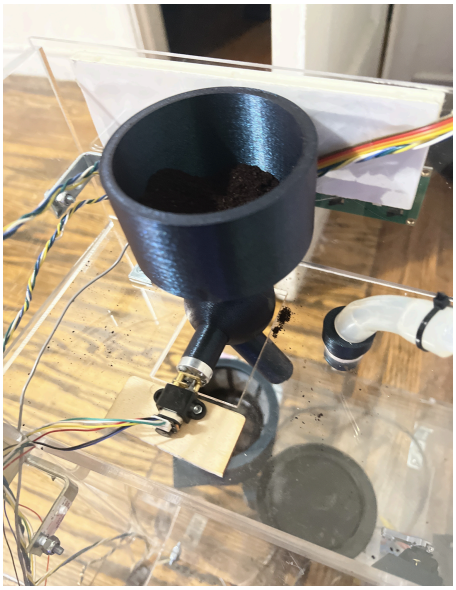


Figure 5. Coffee ground storage and the motor

- Heat insulation around the heating element : Heating element is mounted on the top of one of the acrylic layers attached with heat conductive metal (aluminum) and not touching the bottom acrylic layer, to prevent contact with acrylic or 3D printed materials.
- Water leakage prevention : The water tank was linked with brackets and completely sealed with silicone to prevent water leakage. Between the water tank and bottom layer where circuits and the boards are embedded, there is the extra acrylic layer to prevent the water directly dropping to the circuit if any leakage from the water tank happens.
- Single power source for the entire system : Initially the 3 power source was needed for the coffee machine to operate a 120V AC heating element, 5V DC ESP 32, and 10+V regular motor. Due to the insufficient conditions of the regular motor, such as voltage drop and 3rd power source, we replaced the regular motor to the servo motor (5V), reducing the required number of power sources to 2. In addition, by combining the 5V supply and the heating element wire, we have succeeded in using a single power source to power the entire coffee machine.

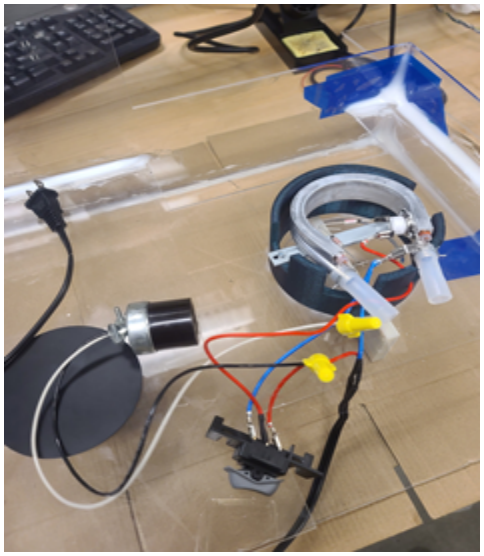


Figure 6. Heating element wire combined to the 5V supply.

2.1.2 Software

- One of the challenges was the coffee dispenser motor.
    - First, we tried stopping the motor after a certain amount of time running at a certain PWM value. However, the speed was not consistent due to frictions and disturbance from coffee grinds.
    - Then, using the PID control model solved the issue, since it gives feedback to PWM value depending on the displacement of each loop. Also we modified that each step is small enough(kp = kd = 1, ki = 0.001).
    - Since the step is small enough, we were able to set a bound to stop the motor(+-10%) once it hits the intended rotation range.
    - As the motor rotated based on rotation, we made a rotation-coffee grind relationship based on experiment.
        - One tablespoon was roughly around 25.
        - Number of cups x tablespoon x 25
- In terms of coding, combining all the libraries was another challenge.
    - Servo and motor libraries had conflicts and the code ‘myservo.write()’ line was not read by the ESP32.
    - By changing the servo library from Servo.h to ESP32Servo.h and the order of motor and servo lines in Void Setup(), we were able to function the code.
        - Regular motor setup had to be written before the servo setup.
- One feature that makes us different is our social media platform. We utilized google sites and Firebase (Powered by google) to upload and send data in HTML onto our website. We created an embedded portion onto the website that sends data that was uploaded to Firebase from our ESP. Once the data is in Firebase, it is displayed into the published website.

3. Procedure

- First, to use the coffee machine, press the buttons to choose the mode.
    - Button 1 for personal coffee.
    - Button 3 for modification
    - Button 2 to continue
- Then, press button 1 or 3 to get right number of cups to brew and check if the value is correct via LCD display
- If button 3 was pressed on the first step, it will require the user to input one more information, the tablespoon number.
    - Again, press button 1 or 3 to adjust the number of tablespoons of coffee grind per cup and press button 2 to continue.
- Now, the coffee dispenser will work and the LCD will display to turn on the heating element switch. Meanwhile the machine will send the data to the website.
- Once brewed, the user can keep the heating element on if one wants the coffee pots to be kept warm.
- If the user used mode 1, personal coffee, press button 1 to strengthen the personal coffee flavor or button 3 to weaken the personal coffee flavor.
- Later, the user can press button 4 twice to initiate the cleaning process.
- During all of these procedures, users can cancel the input by pressing button 4.

4. Engineering Analysis

4.1 Hardware

4.1.1 Heating Element Heat Generation

- For the heating element, we are utilizing 120V from the outlet to power it.

$$P = \frac{V^2}{R} \rightarrow Q = Pt \rightarrow \Delta T = \frac{Q}{mc}$$

  Using this power equation, the power is used to generate heat which results in the rise of temperature.
- With the measured 120V and a resistance of 29 ohms:

$$P = 120^2 / 29 = 590 \text{ watts}$$

- The heating element reaches temperatures around 90-96 degrees celsius, enough for the water to almost begin to boil.
- Sensors and fuses regulate the temperature of the heating element and prevent it from overheating.

4.1.2 Water Transfer

- As the water reaches boiling point, steam starts to generate within the tube. Due to the tube having a small diameter, the bubbles created from the steam are able to carry the water up the tube towards the nozzle where it then drips out into the filter.
- A one way valve is utilized between the water tank and the heating element to prevent the water from flowing back into the tank and ensure that it only travels up the tube towards the nozzle.

4.1.3 Filter Motor

- The coffee filter holder location and weight were utilized to determine the amount of torque created and needed by the motor in order to handle the flipping of the holder..
- Taking in the mass of water + filter in filter holder + coffee grinds = about 2 kg.
    - The lab motor with a stall torque of 1.3 kg/cm would have had issues flipping the filter. As such we decided to use a DS3235 35KG servo motor which has a rated stall torque of 29 kg/cm at 5V (which is the amount we are using to power the motor) which is more than enough to ensure the filter holder has enough force to flip properly.
- The filter holder was placed in a specific location to be able to flip 180 degrees as well as not interfere with the nozzle or the coffee pot while flipping.

4.1.4 Coffee Grind Dispenser

- Our coffee grind dispenser design was inspired by the design of a cereal dispenser. It contains three parts; the top cylindrical housing that holds the grinds, the middle spherical housing that holds the flywheel, and the tube that connects the grinds from the dispenser to the filter.
- The top part of the body contains a cylindrical piece with a funnel at the bottom. By putting the funnel hole in the center of the cylindrical piece, we ensure that all the grinds in the dispenser will correctly flow into the sphere shaped housing.
- The middle part contains the spherical housing. By completely containing the flywheel, we are able to ensure that no coffee grinds leaks out while the wheel is rotating.
- The bottom part is the tube that connects the spherical housing to the filter. By having it at an angle and having it be enclosed, we ensure that the grinds flow correctly and that none of it leaks out.

4.2 Software

4.2.1 Button Logic

- Feedback Coffee features four buttons to measure user input. In order to minimize the amount of buttons needed, we coded so that each button has a different role when the coffee machine is in different phases.

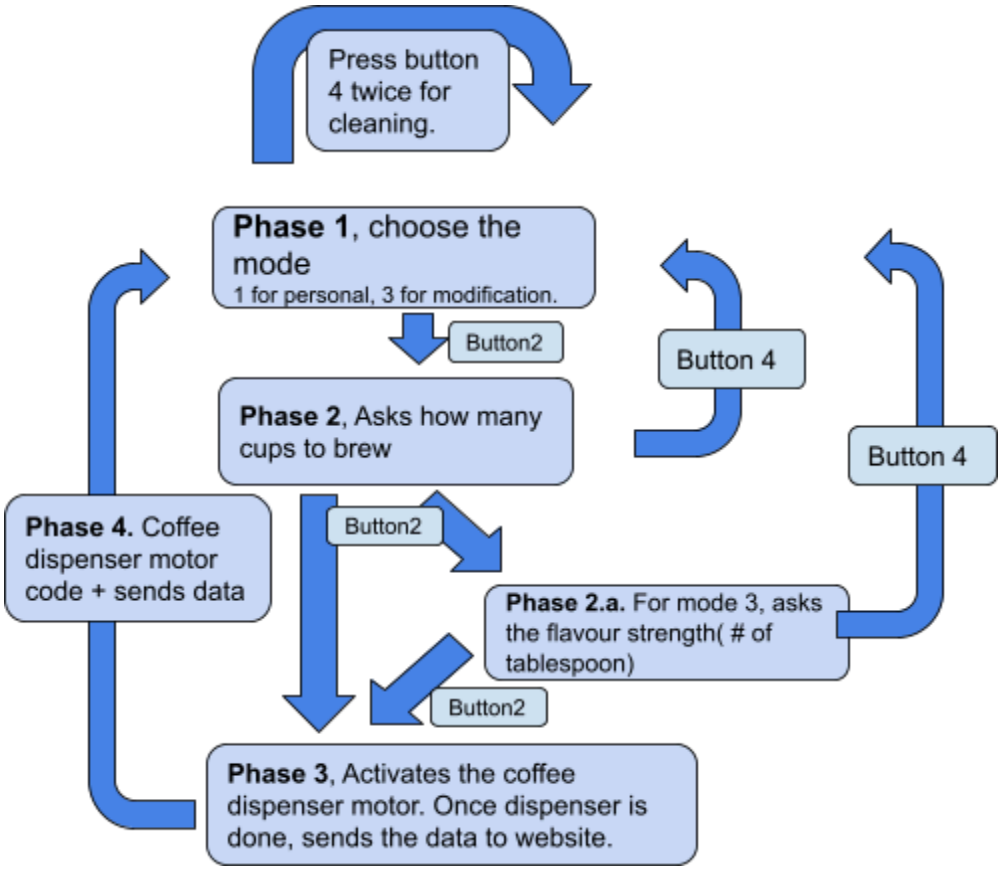


Figure 7. Button Logic Map

- Buttons 1 and 3 are used for changing the different modes that we have implemented, as well as used for adjusting the amount of coffee grind that is put into the filter, during that phase.
- Button 2 is used for confirming your selection, and button 4 is used to cancel and to clean the coffee filter holder.

4.2.2 LCD Display

- Feedback Coffee also features an LCD display in order to communicate with the user as well as display any information that is needed. It will display the current phases the coffee machine is in as well as display the amount of cups of coffee that will be added.
- Since the space for the LCD was limited to 32 characters the display was coded so that the text would slide to the left until it reaches the end of the sentence before it resets.
    - Using millis(), we made a log of current time and last saved time that after a certain amount of time passed, the word will slide to the left by 1 character.

5. Reflection

5.1 Result

- Overall, the final product of Feedback Coffee was a success. All of the features we intended to have from the beginning managed to make it into the final product, and on top of that all the features were working as expected although some more effectively than others.
- The machine successfully takes into consideration the amount of coffee grinds the user desires and brews that amount of coffee. The machine then uploads the data onto the website so that you and your friends can keep track of how much coffee you have drunk.
- The self-cleaning filter and motorized dispenser all successfully create the automated experience we were looking for. The machine also successfully records and stores your feedback for further use.

5.2 Problem & Solution

- The problem that we tackled was that coffee is at the start of most peoples' days, yet no one is monitoring the effects of caffeine and also taking advantage of this commonality that most of us Americans have. We here at Feedback Coffee have taken an American morning ritual and turned it into a social media platform along with raising awareness of the total caffeine intake that we have as per our cup intake.

5.3 What we wish to change

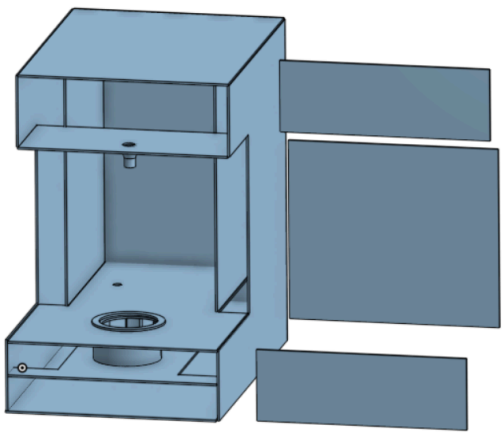
- Building this advanced system with materials from our lab kit was not a very efficient way to integrate this circuit. We had multiple instances where simple wire connections were out of place and they caused errors that had us debugging for hours if not days.
- In a future prototype we would like to use a PCB board manufactured at a fabrication lab. We plan to use KiCad to create the schematic (as shown in 6.3) along with the footprint PCB layout. Then we would export the gerber files to a fab lab and create functioning PCB boards that would house our resistors and components. We could also use a manufacturing distributor like Digikey to buy our components.

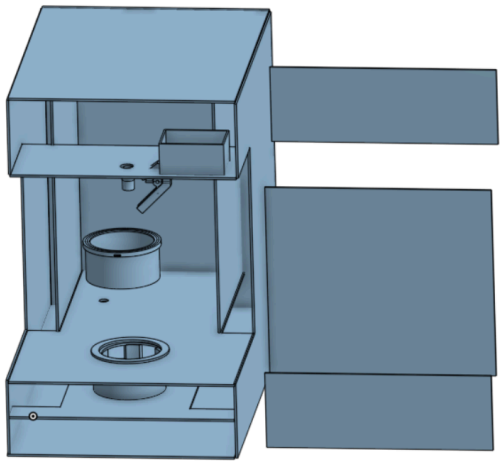
6. Appendix

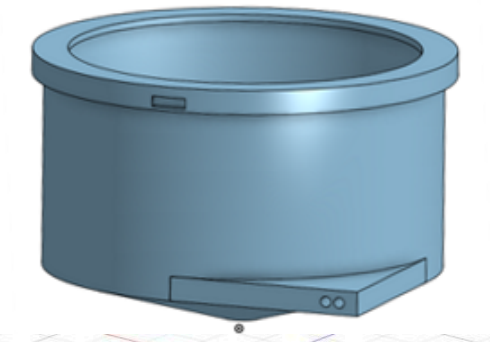
6.1 BOM

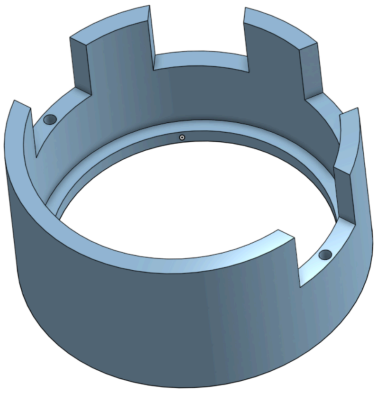
- https://docs.google.com/spreadsheets/d/1iBpz5azHXMAb13UAmqphbY8WRzB9vM8iQSzzrSeeDDw/edit?usp=sharing

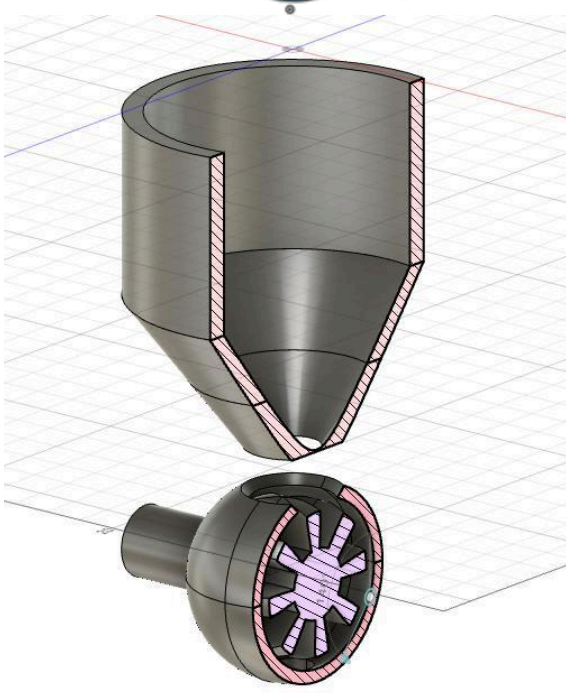
6.2 CAD

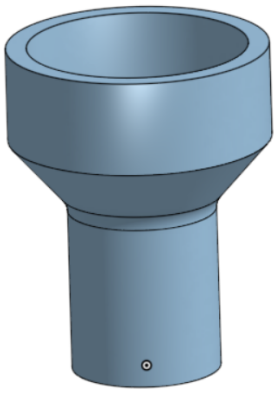




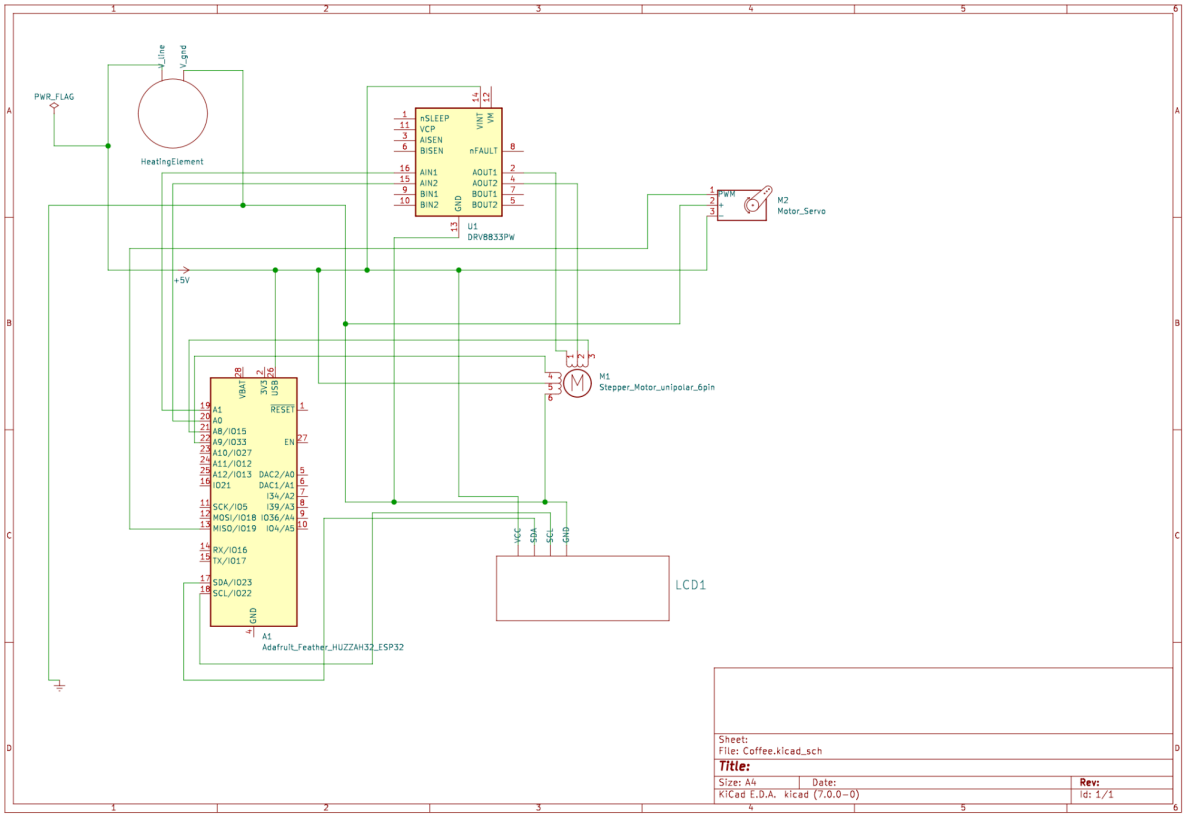








6.3 Circuit Diagram



6.4 Website/Data Tracking

https://sites.google.com/berkeley.edu/feedback-coffee/home

6.5 Code

```
// Define the pin numbers for the buttons
#include <ESP32Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Arduino.h>
#include <ESP32Encoder.h>
#include <WiFi.h>
#include <FirebaseESP32.h>

#define BIN_1 26
#define BIN_2 25
// the wifi and the Firebase link stuff
#define WIFI_SSID "hrm553"
#define WIFI_PASSWORD "becoming553"
#define FIREBASE_HOST "feedbackcoffee-a7f8c-default-rtdb.firebaseio.com" // Your Firebase Realtime Database URL
#define FIREBASE_AUTH "your_firebase_secret" // Your Firebase database secret

ESP32Encoder encoder;
FirebaseData firebaseData;

unsigned long curtime;
```

```
static unsigned long intertime = 0;

Servo myservo;

#define BUTTON1_PIN 17  // the first button " + "
#define BUTTON2_PIN 16  // the 2nd button "confirm "
#define BUTTON3_PIN 19  // the 3rd button " - "
#define BUTTON4_PIN 14  // the 4th button "reset" + "cleanig"
int servoPin = 21;
// Set the LCD number of columns and rows
int lcdColumns = 16;
int lcdRows = 2;
int test = 0;

/* setting PWM properties */
const int freq = 5000;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int resolution = 8;
int MAX_PWM_VOLTAGE = 200;
unsigned long time_now = 0;
int laststep = 0;

float kp = 1;
float ki = 0.001;
float kd = 1;
float  gain;
float kv = 0.15;
float ds = 0;
float pp;
float p;
float y_0 = 0.1;
float y_e;
float e;
float esum = 0;
float yi;
float antiI = 0.1;
float dp=0;
float de;
float yd;
float pose;
float posesum = 0;
float v = 0;
float p_last = 0;
const long dt = 20;
float e_last = 0 ;
float dp_last = 0;
int mod1TS = 2;
int feedbackcheck = 0;
unsigned long  checktime;

// Set the I2C pins
const int SDA_PIN = 23;
const int SCL_PIN = 22;

// Set LCD address, number of columns and rows
// If you don't know your display address, run an I2C scanner sketch
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);

int pos = 0;
int modcheck = 0; // 0 as the server, 1 as personal
int cupnum = 1;
int defaultrot = 3; // lets say 3 is the default for 2table spoon
int totalrot; // total rotation mulipled default * cupnum
float TS = 2;
int grindready =0;
int cupsBrewed = 0;
int caffeineIntake = 0;
int dataready;
int cup;
int caffeine;

void setup() {
  // Initialize serial communication at 9600 baud rate
  Serial.begin(115200);

  ESP32Encoder::useInternalWeakPullResistors = UP; // Enable the weak pull up resistors
  encoder.attachHalfQuad(27, 33); // Attache pins for use as encoder pins
  encoder.setCount(0);  // set starting count value after attaching

  /* configure LED PWM functionalitites */
  ledcSetup(ledChannel_1, freq, resolution);
  ledcSetup(ledChannel_2, freq, resolution);

  /* attach the channel to the GPIO to be controlled */
  ledcAttachPin(BIN_1, ledChannel_1);
  ledcAttachPin(BIN_2, ledChannel_2);

  myservo.setPeriodHertz(50);
  myservo.attach(servoPin, 400,2400);
```

```
  // Set the button pins as inputs with pullup resistor enabled
  pinMode(BUTTON1_PIN, INPUT_PULLUP);
  pinMode(BUTTON2_PIN, INPUT_PULLUP);
  pinMode(BUTTON3_PIN, INPUT_PULLUP);
  Wire.begin(SDA_PIN, SCL_PIN);
  // Initialize LCD
  lcd.init();
  // Turn on LCD backlight
  lcd.backlight();


  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

}
// button 1 is minus also mode 1
// button 2 is confirm
// button 3 is plus  also mode 3
void loop() {
  curtime = millis();
  if (laststep == 0){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("1 for personal");
    lcd.setCursor(0,1);
    lcd.print( test + "3 for modification");
    if(curtime - intertime > 500){
      test += 1;
      intertime = curtime;
    }
    if (test == 5){
      test = 0;
    }

  }
  if (laststep == 1){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(test + "press the switch");
    lcd.setCursor(0,1);
    lcd.print(test + "to start brewing");
    if(curtime - intertime > 600){
      test += 1;
      intertime = curtime;
    }
    if (test == 8){
      test = 0;
    }
    if (feedbackcheck ==0){
      checktime = curtime;
      feedbackcheck = 1;

    }
    if (curtime - checktime > 30000){
      laststep = 2;
      test = 0;
      feedbackcheck = 0;
    }

  }
  if(laststep == 2){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print( test + "press 1 for more flavor");
    lcd.setCursor(0,1);
    lcd.print( test + "press 3 for less flavor");
    if(curtime - intertime > 600){
      test += 1;
      intertime = curtime;
    }
    if (test == 10){
      test = 0;
    }
    if (feedbackcheck ==0){
      checktime = curtime;
      feedbackcheck = 1;

    }
    if (curtime - checktime  > 30000){
      laststep = 1;
      test = 0;
      feedbackcheck = 0;
```

```
  }
 }


 // Check if Button 1 is pushed
 if (digitalRead(BUTTON1_PIN) == LOW) {
   // Print a message to the serial monitor
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("Button 1 is pushed!");
   Serial.println("Button 1 is pushed!");
   if((modcheck == 1)&&(laststep != 0)){
     mod1TS = mod1TS + 0.5;
   }
   modcheck = 1; // one for the personal
   laststep = 0;
   delay(300); // Add a delay to debounce the button
 }

 // Check if Button 3 is pushed
 if (digitalRead(BUTTON3_PIN) == LOW) {
   // Print a message to the serial monitor
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("Button 3 is pushed!");
   Serial.println("Button 3 is pushed!");
   if((modcheck == 1)&&(laststep != 0)){
     mod1TS = mod1TS + 0.5;
   }
   modcheck = 3;  // 3 for the modification
   laststep = 0;
   delay(300); // Add a delay to debounce the button
 }

 if (digitalRead(BUTTON4_PIN) == LOW) {
   // Print a message to the serial monitor
   delay(300); // Add a delay to debounce the button
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("Button 4 is pushed!");
   Serial.println("Button 4 is pushed!");
   laststep = 0;
   if (digitalRead(BUTTON4_PIN) == LOW) {
     // Print a message to the serial monitor
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("cleaning");
     Serial.println("cleaning");
     myservo.write(120);
     delay(1000);
     for(int i=0; i < 5 ; i +=1 ){
       myservo.write(110);
       delay(400);
       myservo.write(120);
       delay(400);
     }
     myservo.write(0);
     delay(1000); // Add a delay to debounce the button
   }
   delay(300);
 }
 // Check if Button 2 is pushed meaning the confirm is pressed after choosing the mode
 if (digitalRead(BUTTON2_PIN) == LOW) {
   delay(300);
   laststep = 0;
   while (true){
     Serial.println("showing screen 2, current cups of coffee = " + String(cupnum));
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print(String(cupnum) + " cups");
     lcd.setCursor(0,1);
     lcd.print("1 = +, 3 = -");
     if (digitalRead(BUTTON1_PIN) == LOW){
       cupnum += 1;  // button 1 for minus
       if (cupnum <1){
         cupnum = 1;
       }
       delay(300);
     }
     if (digitalRead(BUTTON3_PIN) == LOW){
       cupnum -= 1;  // button 3 for plus
       if (cupnum>5){
         cupnum =5;
       }
       delay(300);
     }
     if (digitalRead(BUTTON4_PIN) == LOW){
       break;
       break;
       break;
     }
```

```
  if (digitalRead(BUTTON2_PIN) == LOW){
    delay(300);
    if (modcheck == 1){
      Serial.println("screen 3.1 - turn on the switch for brewing"); // showing turn on switch for the personal coffee
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Turn on the switch");
      lcd.setCursor(0,1);
      lcd.print("for brewing");
      delay(300);
      if (digitalRead(BUTTON4_PIN) == LOW){
        break;
        break;
      }
      grindready = 1;
    }
    else if (modcheck == 3){
      while (true) {
        Serial.println("screen 3.2 + showing # of tablespoon" + String(TS));
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print(String(TS) + " tablespoon");
        lcd.setCursor(0,1);
        lcd.print("1 = +, 3 = -");
        if (digitalRead(BUTTON1_PIN) == LOW){
          TS += 0.5;
          if (TS < 1){
            TS = 1;
          }
          delay(300);
        }
        if (digitalRead(BUTTON3_PIN) == LOW){
          TS -= 0.5;
          if (TS > 8){
            TS = 8;
          }
          delay(300);
        }
        if (digitalRead(BUTTON4_PIN) == LOW){
          break;
          break;
          break;break;
        }
        if (digitalRead(BUTTON2_PIN) == LOW){
          grindready = 1;
          break;
        }
      }

    }

  }
  if (grindready == 1){
    break;
  }
}
if (grindready == 1){
  Serial.print("grindmotor code");
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("preparing");
  lcd.setCursor(0,1);
  lcd.print("coffee grind");
  grindready = 0;
  if (modcheck == 1){
    TS = mod1TS;
  }
  dp = cupnum * TS * 25+ dp_last;
  TS = 2;
  while (true){
    time_now = millis();
    p = encoder.getCount() / (75.81 * 6) * 2 * M_PI;
    int pwmzero = 0;

    /* motor sine wave section */
    if ((  dp *0.9 < p)&&(p < dp * 1.1)){
      MAX_PWM_VOLTAGE =0;
      pwmzero = 1;
    }
    float output =  MAX_PWM_VOLTAGE;

    if (output > 0) {
      ledcWrite(ledChannel_1, output);
      ledcWrite(ledChannel_2, LOW);
    }
    else {
      ledcWrite(ledChannel_2, -output);
      ledcWrite(ledChannel_1, LOW);
    }
    pp = ( (p - p_last) / (dt*0.001)); // pp as actual speed

    pose = dp - p; // position error desired position - counted position
```

```
      posesum = posesum + pose; // position error sum

      yd = pp * kd;

      if (p != dp){
        MAX_PWM_VOLTAGE = MAX_PWM_VOLTAGE + kp*pose + ki * posesum - kd * pp;
      }
      Serial.print(p);


      p_last = p  ;
      if (pwmzero == 1){
        break;
      }



      while (millis() < (time_now + dt)) {};
    }
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Coffee grind");
    lcd.setCursor(0,1);
    lcd.print("Ready");
    grindready = 0;
    laststep = 1;
    dp_last = dp;
    dataready = 1;
    cupsBrewed = cupsBrewed + cupnum;
    caffeineIntake = caffeinIntake + cupnum * TS /2 * 120;
   }


   delay(900);
  }

  if (dataready == 1){
     if (Firebase.setInt(firebaseData, "/cupsBrewed", cupsBrewed)) {
       Serial.println("Cups Brewed data was sent to Firebase");
       cup = 1;
     } else {
       Serial.println("Failed to send Cups Brewed data to Firebase");
       Serial.println("Error: " + firebaseData.errorReason());
     }

     if (Firebase.setInt(firebaseData, "/caffeineIntake", caffeineIntake)) {
       Serial.println("Caffeine Intake data was sent to Firebase");
       caffein = 1;
     } else {
       Serial.println("Failed to send Caffeine Intake data to Firebase");
       Serial.println("Error: " + firebaseData.errorReason());
     }
  }
  if ((cup ==1 ) && (caffein = 1 )){
    dataready = 0;

  }

}
```

Firebase Data Embedded (HTML)

```
<div id="cupsBrewed"></div>
<div id="caffeineIntake"></div>
<script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-database.js"></script>
<script>
  var firebaseConfig = {
    apiKey: "AIzaSyDnc76REKppd3OP3GmMB6_RvWxdKTAv2pU",
    authDomain: "feedbackcoffee-a7f8c.firebaseapp.com",
    databaseURL: "https://feedbackcoffee-a7f8c-default-rtdb.firebaseio.com",
    projectId: "feedbackcoffee-a7f8c",
    storageBucket: "feedbackcoffee-a7f8c.appspot.com",
    messagingSenderId: "154854701444",
```

```
    appId: "1:154854701444:web:b0d638ea64bead0f3385d1",
    measurementId: "G-JRRYDJ7V79"
  };

  firebase.initializeApp(firebaseConfig);

  firebase.database().ref('/cupsBrewed').on('value', function(snapshot) {
    document.getElementById('cupsBrewed').innerHTML = "Cups Brewed: " + snapshot.val();
  });

  firebase.database().ref('/caffeineIntake').on('value', function(snapshot) {
    document.getElementById('caffeineIntake').innerHTML = "Caffeine Intake: " + snapshot.val() + "mg";
  });
</script>
```