TeaMaker

ME 102B Group 7: Harrison Lee, Franklin Ho, Alahe Akhavan, David Lu

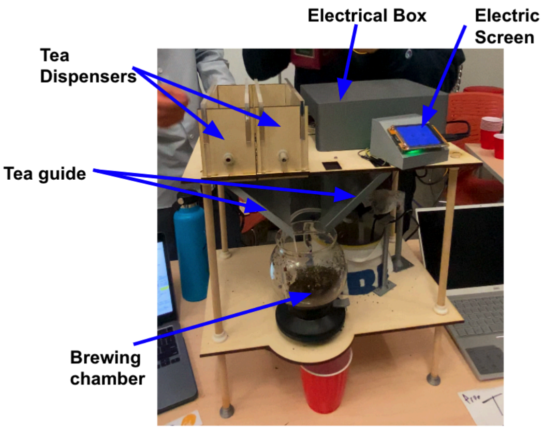
Opportunity:

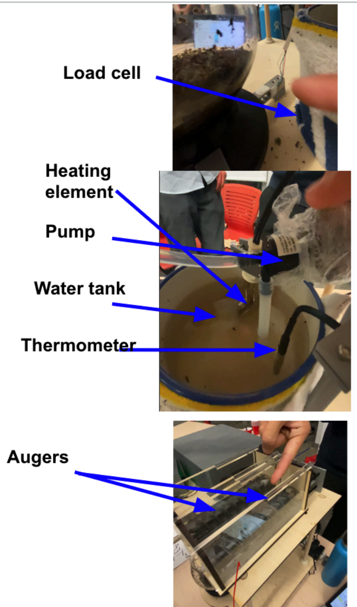
Many people choose to drink tea instead of coffee in the morning. We would like to make an automated tea maker for Chinese Tea, which calls for different water temperatures, volumes and brewing times for each unique type of tea. Users will also be able to produce tea straight from the tea leaves without needing to buy expensive pods (eg. Nespresso) found in conventional coffee machines. Similar solutions online include a Keurig (automated coffee maker), fast-food soda dispensers (dispenser mechanism), Ember mug (temperature control of a coffee mug).

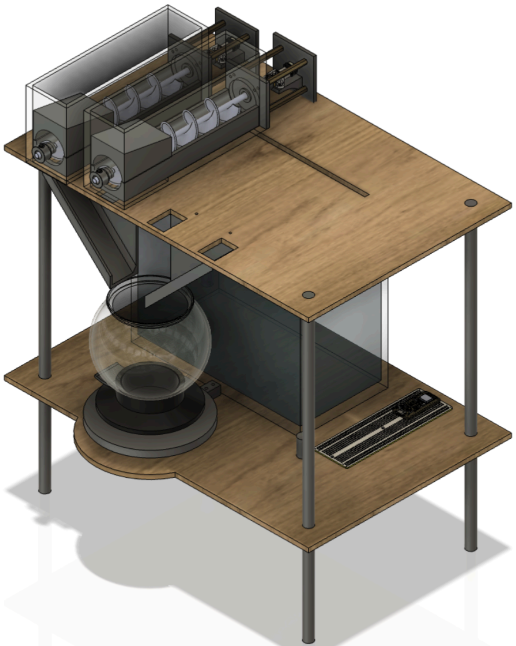
High Level Strategy:

Our traditional tea making system includes 5 subsystems - tea selection using touch screen, tea dispensing with DC motor, load cell, water heating, and water pump. Every subsystem is controlled via one main ESP32 microcontroller. To use this device, the user first selects between the two tea options via touch screen. Options available on the touch screen include 1) green tea, 2) black tea, 3) refill with water, and 4) reset. After having selected the tea option, the tea box dispenses tea using the transmission system, the auger. While the tea is being dispensed in the brewing chamber, the load cell actively measures the weights of the tea leaves and gives feedback to the transmission system to stop once the goal weight of the brewing chamber is reached. Then the thermometer reads the temperature of the water, such that the heating element is not exceeding the goal temperature. Once the water temperature is at the desired temperature (80-90C depending on tea type), and the tea is in the brewing chamber, the pump begins deposing the hot water in the brewing chamber. Tea is served after the wait time for brewing is completed. Initially, in addition to the final design, our goal was to have 4 tea selection types, with a reservoir of cold water fed into a smaller hot water reservoir for our water heater design. Given financial and time constraints, we decided to move forward with two tea dispensing boxes and adjust to a smaller, singular water tank. This did not limit our final prototype. A concern of ours was food safety. We addressed this by using food safe material for the systems which contacted hot water, such as using a commercial grade tea brewing chamber and pump. We were able to successfully achieve what we set out to initially in terms of device functionality, mechanical stability, and safety.

Integrated Physical Device:







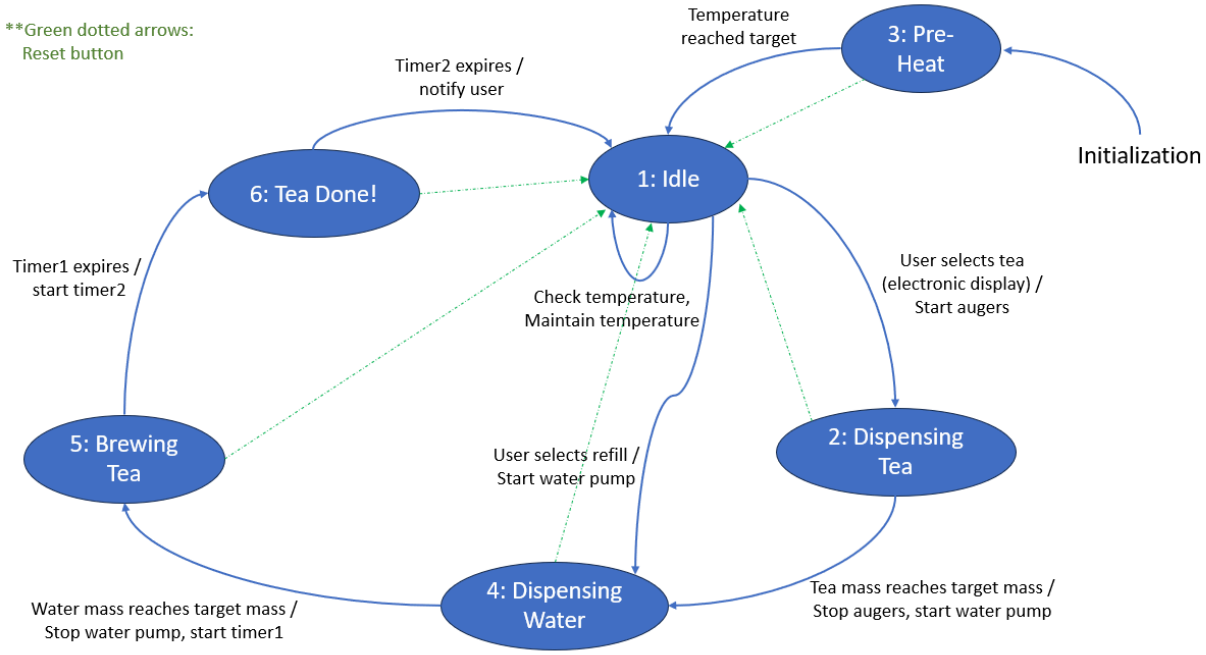
Critical Design Dimensions/Calculations:

Our selected motor was a 210:1 6V Pololu Brushed DC Motor, with a maximum stall torque of 3.0kg · cm. We calculated the expected torque by manufacturing a first prototype of our auger with a handle of length 5cm that we could turn manually. We filled the box with tea leaves and used a spring force sensor to determine the force on the 5cm-long handle required to produce enough torque to rotate the auger full of leaves. We picked this motor with a safety factor of 3.53 to give us an optimal balance between required torque and actuation speed (rpms). This safety factor was crucial as it later turned out that leaves would get stuck in the auger over time, requiring a greater than expected torque. Our transmission experienced very small axial forces. To calculate the maximum transmission force from a full box of leaves, we calculated the highest mass flow rate (g/s) of leaves, multiplied by its horizontal velocity in the auger (cm/s) to determine axial force.

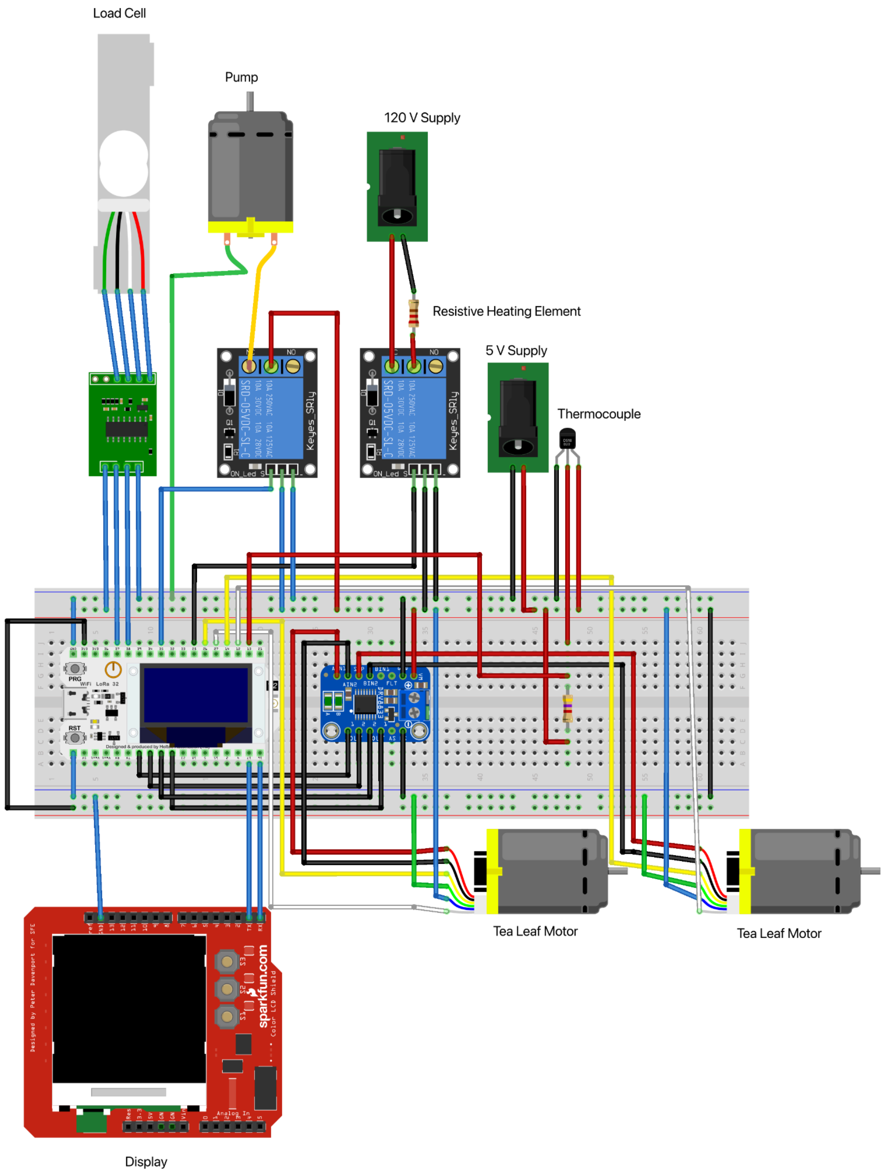
Required torque = 5cm * 1N = 5N · cm = 0.51kg · cm
Safety factor = 3/0.51 * 60% (safety factor for sustained max torque) = 3.53

Axial force on transmission system:
= Rate of change of momentum of tea leaves
= 100g*8cm/s per second = 800g · cm/s^2 = 0.008kg · m/s^2= 0.008N

State Transition Diagrams:


**Green dotted arrows:
Reset button
Temperature reached target
3: Pre-Heat
Initialization
Timer2 expires / notify user
1: Idle
6: Tea Done!
Timer1 expires / start timer2
Check temperature, Maintain temperature
User selects tea (electronic display) / Start augers
5: Brewing Tea
User selects refill / Start water pump
2: Dispensing Tea
4: Dispensing Water
Water mass reaches target mass / Stop water pump, start timer1
Tea mass reaches target mass / Stop augers, start water pump

Circuit Diagrams:

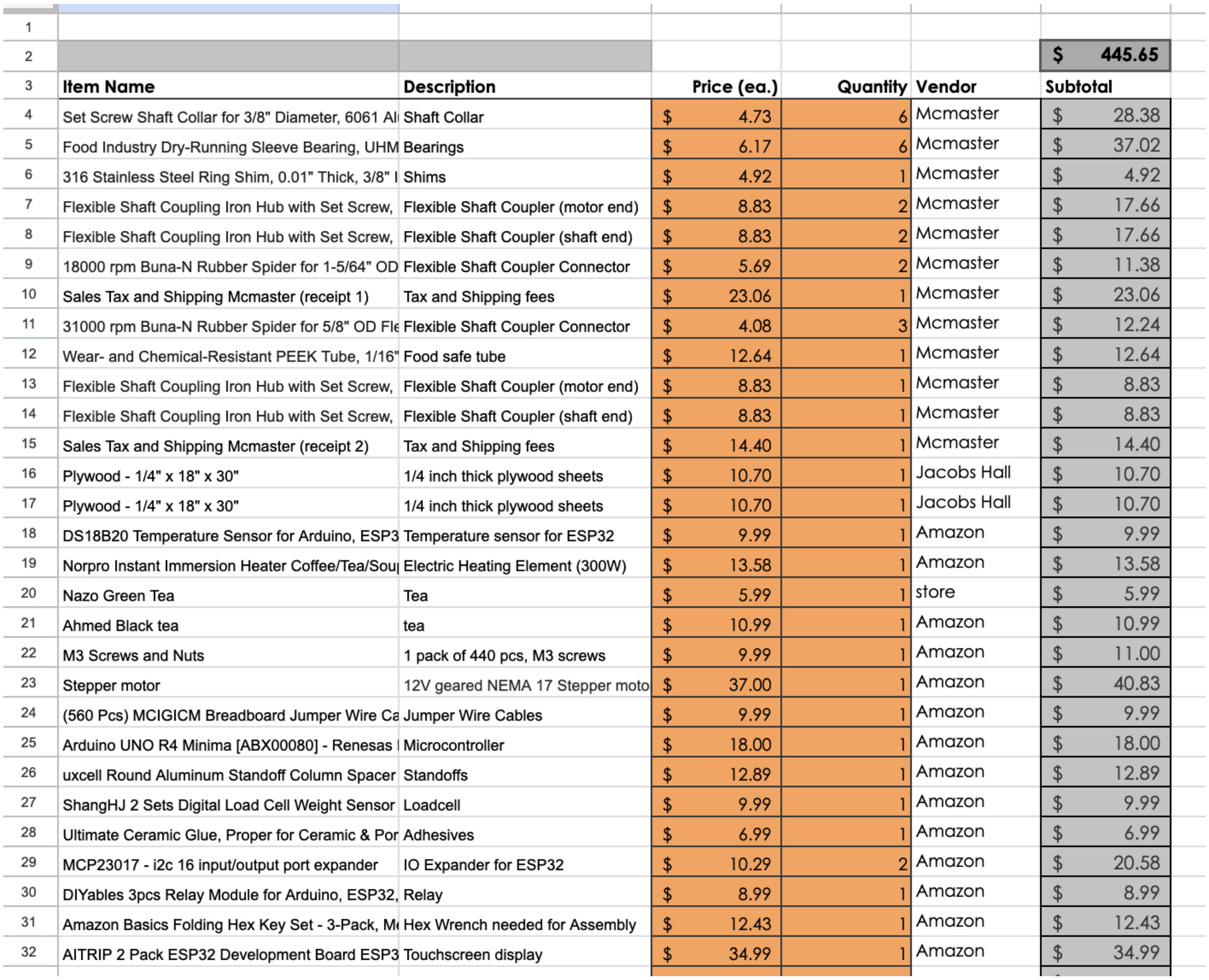


Final Thoughts/Reflection:

From the beginning of the semester, all our team members were dedicated and set on creating a well functioning device, this included good communication, well established roles, and always being present for the weekly meetings. We all were clear on our individual areas of strength and weaknesses, and how to set realistic individual and team goals in which we could achieve by the end of the semester. Some good takeaways for future students - make weekly check-ins and make sure to attend every or share progress, and get as much feedback as you can from the teaching team or staff, this will be crucial for making a good quality system.

Appendices

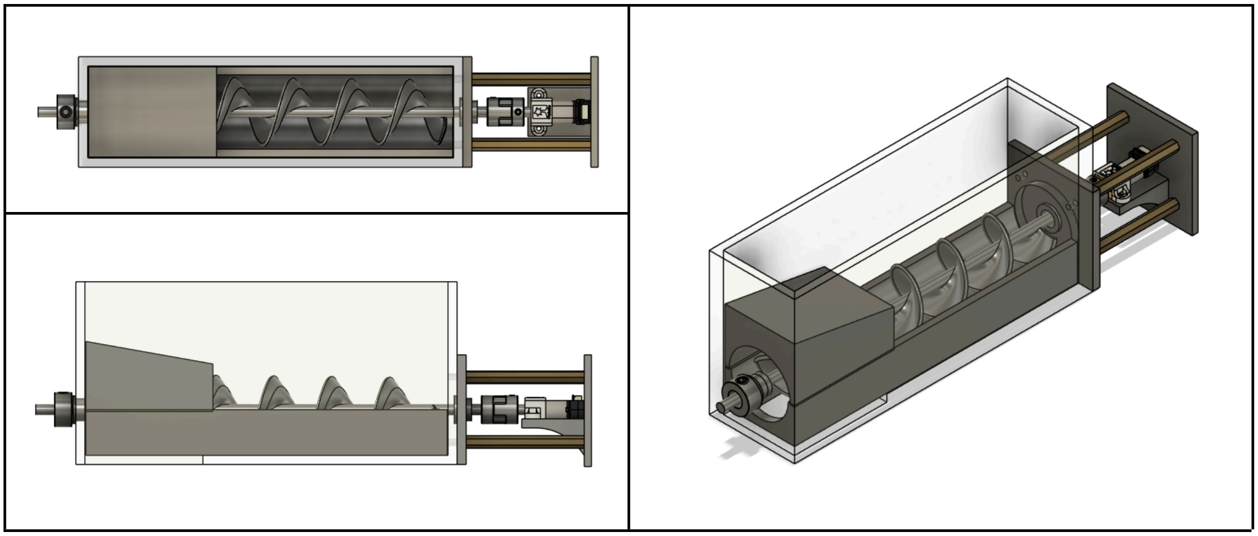
Appendix A: Bill of Materials



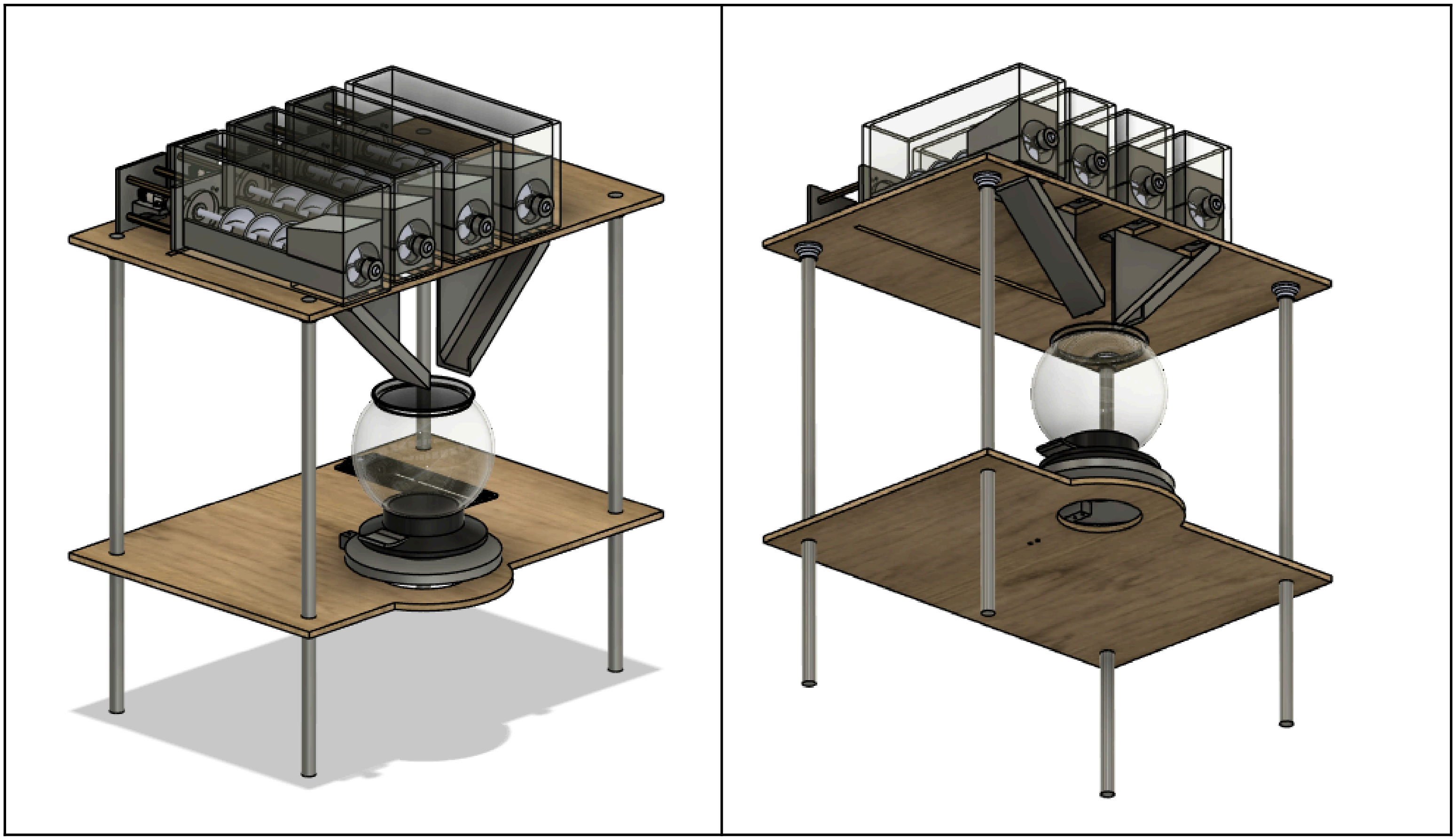
1						
2						$ 445.65
3	Item Name	Description	Price (ea.)	Quantity	Vendor	Subtotal
4	Set Screw Shaft Collar for 3/8" Diameter, 6061 Al	Shaft Collar	$ 4.73	6	Mcmaster	$ 28.38
5	Food Industry Dry-Running Sleeve Bearing, UHM	Bearings	$ 6.17	6	Mcmaster	$ 37.02
6	316 Stainless Steel Ring Shim, 0.01" Thick, 3/8" I	Shims	$ 4.92	1	Mcmaster	$ 4.92
7	Flexible Shaft Coupling Iron Hub with Set Screw,	Flexible Shaft Coupler (motor end)	$ 8.83	2	Mcmaster	$ 17.66
8	Flexible Shaft Coupling Iron Hub with Set Screw,	Flexible Shaft Coupler (shaft end)	$ 8.83	2	Mcmaster	$ 17.66
9	18000 rpm Buna-N Rubber Spider for 1-5/64" OD	Flexible Shaft Coupler Connector	$ 5.69	2	Mcmaster	$ 11.38
10	Sales Tax and Shipping Mcmaster (receipt 1)	Tax and Shipping fees	$ 23.06	1	Mcmaster	$ 23.06
11	31000 rpm Buna-N Rubber Spider for 5/8" OD Fle	Flexible Shaft Coupler Connector	$ 4.08	3	Mcmaster	$ 12.24
12	Wear- and Chemical-Resistant PEEK Tube, 1/16"	Food safe tube	$ 12.64	1	Mcmaster	$ 12.64
13	Flexible Shaft Coupling Iron Hub with Set Screw,	Flexible Shaft Coupler (motor end)	$ 8.83	1	Mcmaster	$ 8.83
14	Flexible Shaft Coupling Iron Hub with Set Screw,	Flexible Shaft Coupler (shaft end)	$ 8.83	1	Mcmaster	$ 8.83
15	Sales Tax and Shipping Mcmaster (receipt 2)	Tax and Shipping fees	$ 14.40	1	Mcmaster	$ 14.40
16	Plywood - 1/4" x 18" x 30"	1/4 inch thick plywood sheets	$ 10.70	1	Jacobs Hall	$ 10.70
17	Plywood - 1/4" x 18" x 30"	1/4 inch thick plywood sheets	$ 10.70	1	Jacobs Hall	$ 10.70
18	DS18B20 Temperature Sensor for Arduino, ESP3	Temperature sensor for ESP32	$ 9.99	1	Amazon	$ 9.99
19	Norpro Instant Immersion Heater Coffee/Tea/Sou	Electric Heating Element (300W)	$ 13.58	1	Amazon	$ 13.58
20	Nazo Green Tea	Tea	$ 5.99	1	store	$ 5.99
21	Ahmed Black tea	tea	$ 10.99	1	Amazon	$ 10.99
22	M3 Screws and Nuts	1 pack of 440 pcs, M3 screws	$ 9.99	1	Amazon	$ 11.00
23	Stepper motor	12V geared NEMA 17 Stepper moto	$ 37.00	1	Amazon	$ 40.83
24	(560 Pcs) MCIGICM Breadboard Jumper Wire Ca	Jumper Wire Cables	$ 9.99	1	Amazon	$ 9.99
25	Arduino UNO R4 Minima [ABX00080] - Renesas	Microcontroller	$ 18.00	1	Amazon	$ 18.00
26	uxcell Round Aluminum Standoff Column Spacer	Standoffs	$ 12.89	1	Amazon	$ 12.89
27	ShangHJ 2 Sets Digital Load Cell Weight Sensor	Loadcell	$ 9.99	1	Amazon	$ 9.99
28	Ultimate Ceramic Glue, Proper for Ceramic & Por	Adhesives	$ 6.99	1	Amazon	$ 6.99
29	MCP23017 - i2c 16 input/output port expander	IO Expander for ESP32	$ 10.29	2	Amazon	$ 20.58
30	DIYables 3pcs Relay Module for Arduino, ESP32,	Relay	$ 8.99	1	Amazon	$ 8.99
31	Amazon Basics Folding Hex Key Set - 3-Pack, M	Hex Wrench needed for Assembly	$ 12.43	1	Amazon	$ 12.43
32	AITRIP 2 Pack ESP32 Development Board ESP3	Touchscreen display	$ 34.99	1	Amazon	$ 34.99

Appendix B: CAD

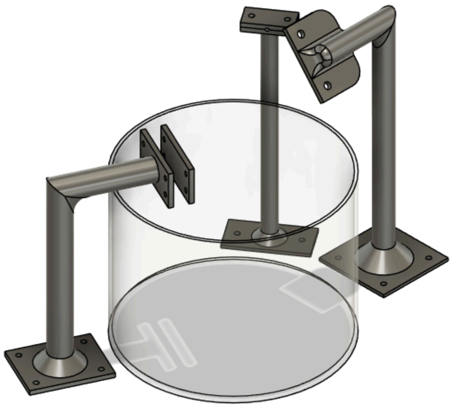
Tea Box with Auger and DC Motor:



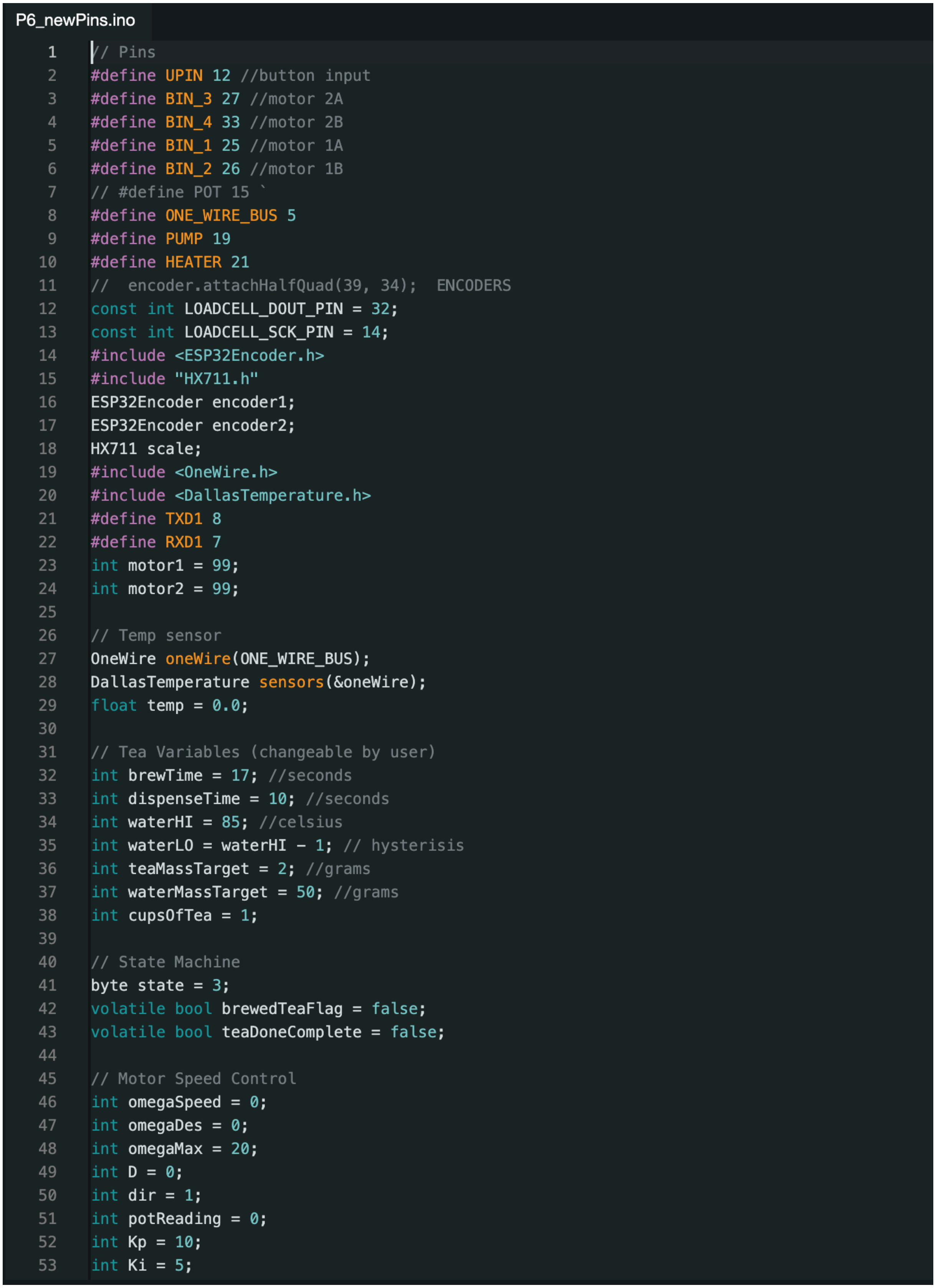
Closer examination of the table structure, with ramps, interfaces with legs, and holes for tea:



Water heater subsystem with mounts for resistive heating element, thermometer and pump:

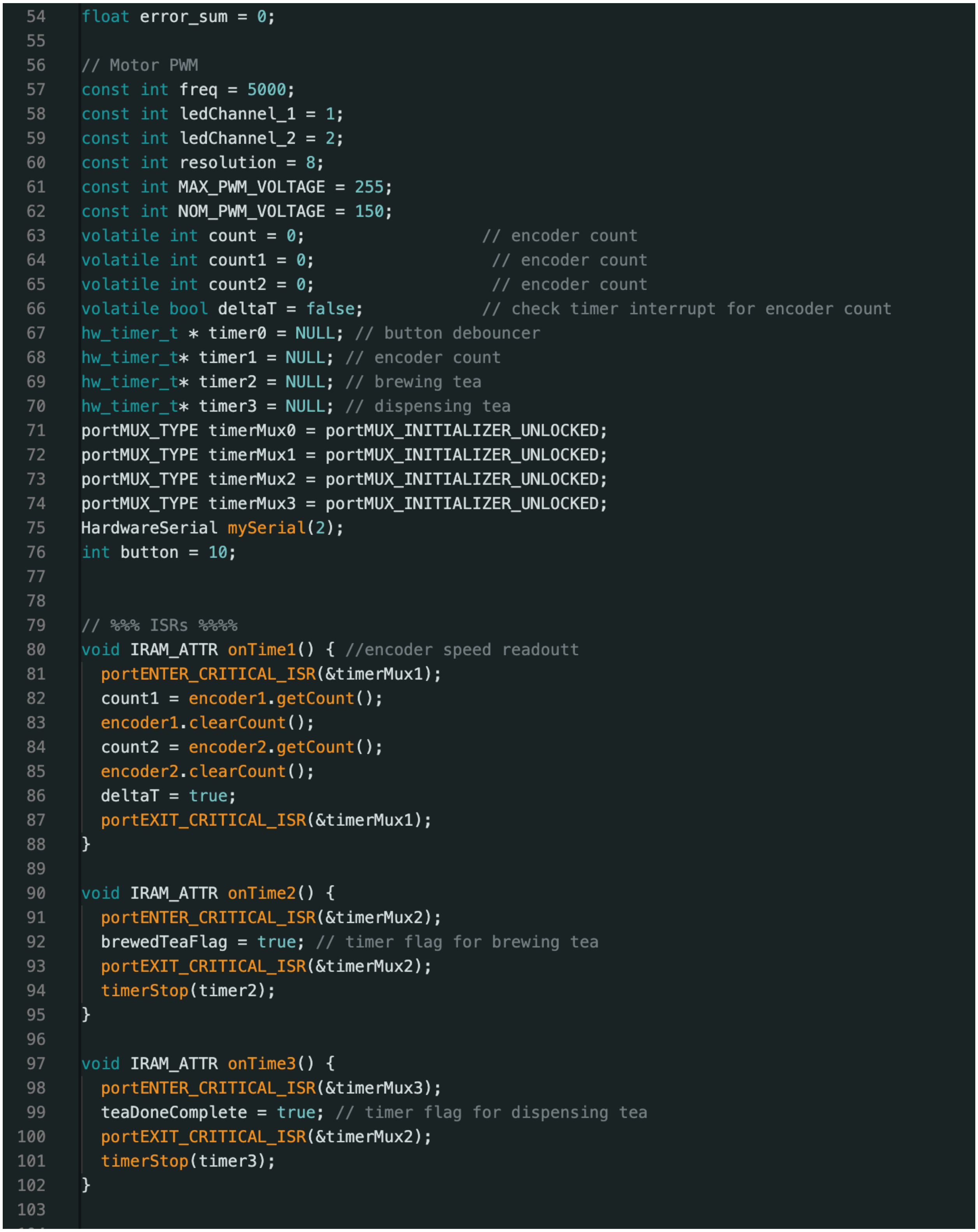


Appendix C: Event-Driven Programming (Arduino IDE)

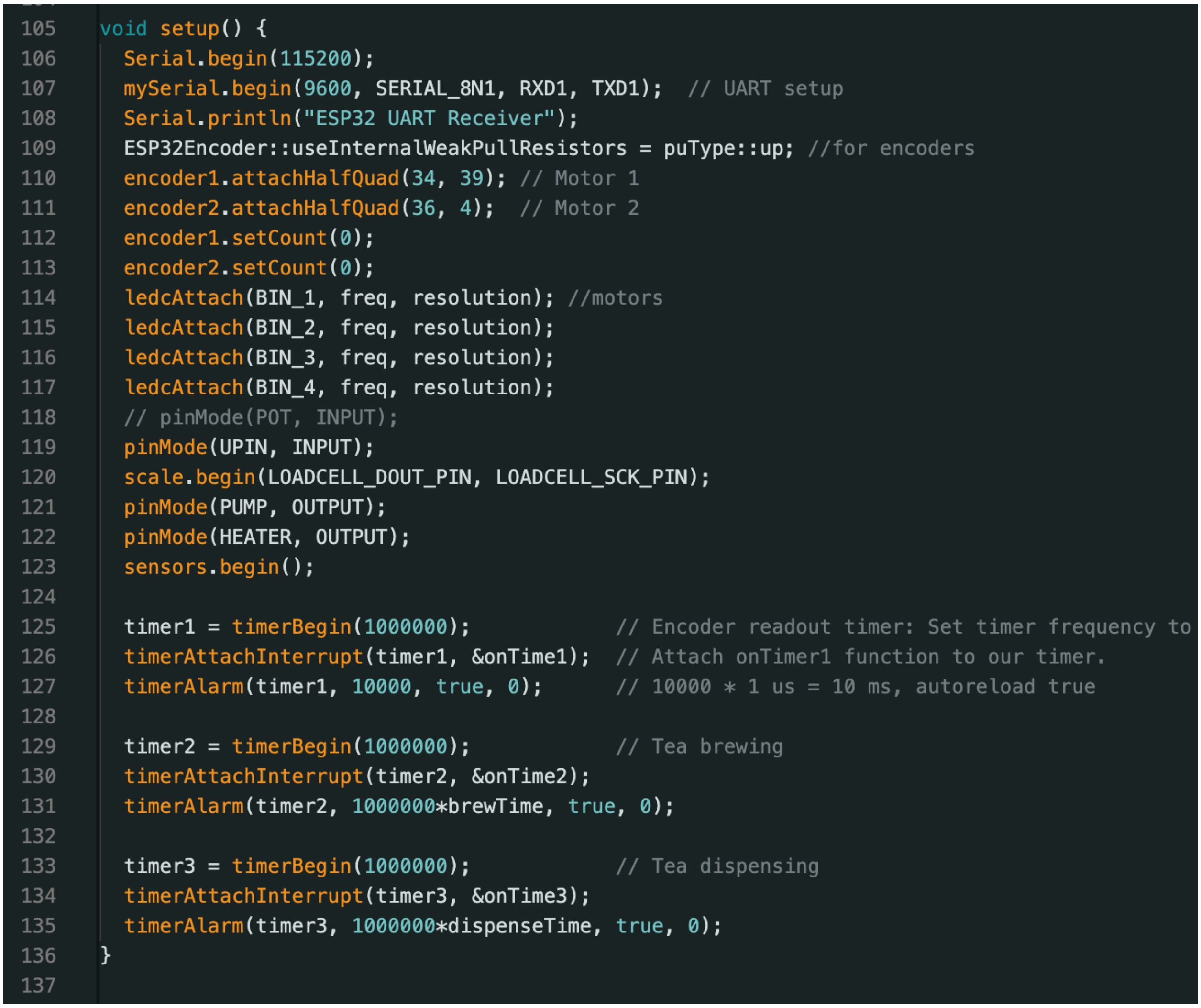


P6_newPins.ino

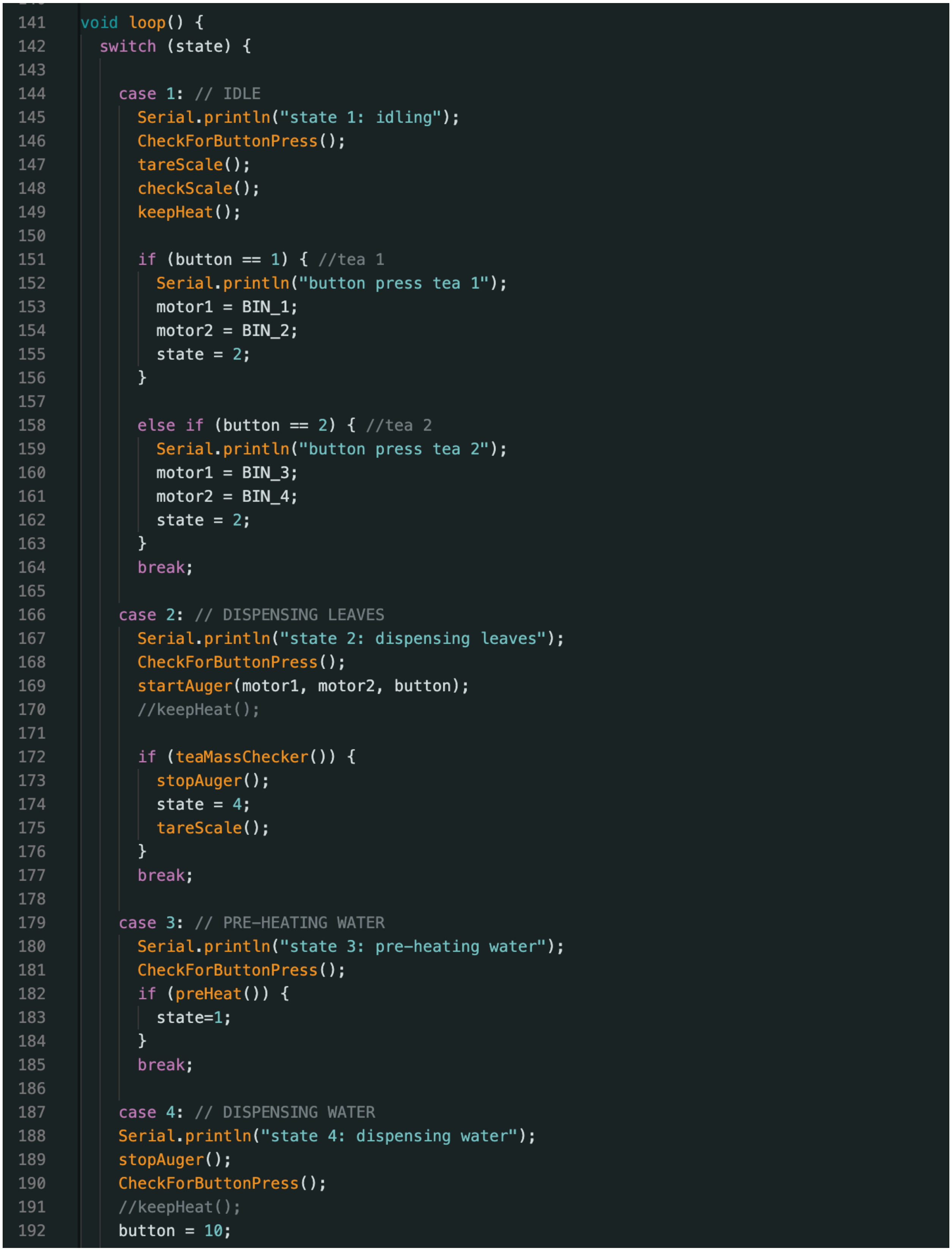
```
// Pins
#define UPIN 12 //button input
#define BIN_3 27 //motor 2A
#define BIN_4 33 //motor 2B
#define BIN_1 25 //motor 1A
#define BIN_2 26 //motor 1B
// #define POT 15 `
#define ONE_WIRE_BUS 5
#define PUMP 19
#define HEATER 21
//  encoder.attachHalfQuad(39, 34);  ENCODERS
const int LOADCELL_DOUT_PIN = 32;
const int LOADCELL_SCK_PIN = 14;
#include <ESP32Encoder.h>
#include "HX711.h"
ESP32Encoder encoder1;
ESP32Encoder encoder2;
HX711 scale;
#include <OneWire.h>
#include <DallasTemperature.h>
#define TXD1 8
#define RXD1 7
int motor1 = 99;
int motor2 = 99;

// Temp sensor
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
float temp = 0.0;

// Tea Variables (changeable by user)
int brewTime = 17; //seconds
int dispenseTime = 10; //seconds
int waterHI = 85; //celsius
int waterLO = waterHI - 1; // hysterisis
int teaMassTarget = 2; //grams
int waterMassTarget = 50; //grams
int cupsOfTea = 1;

// State Machine
byte state = 3;
volatile bool brewedTeaFlag = false;
volatile bool teaDoneComplete = false;

// Motor Speed Control
int omegaSpeed = 0;
int omegaDes = 0;
int omegaMax = 20;
int D = 0;
int dir = 1;
int potReading = 0;
int Kp = 10;
int Ki = 5;
```



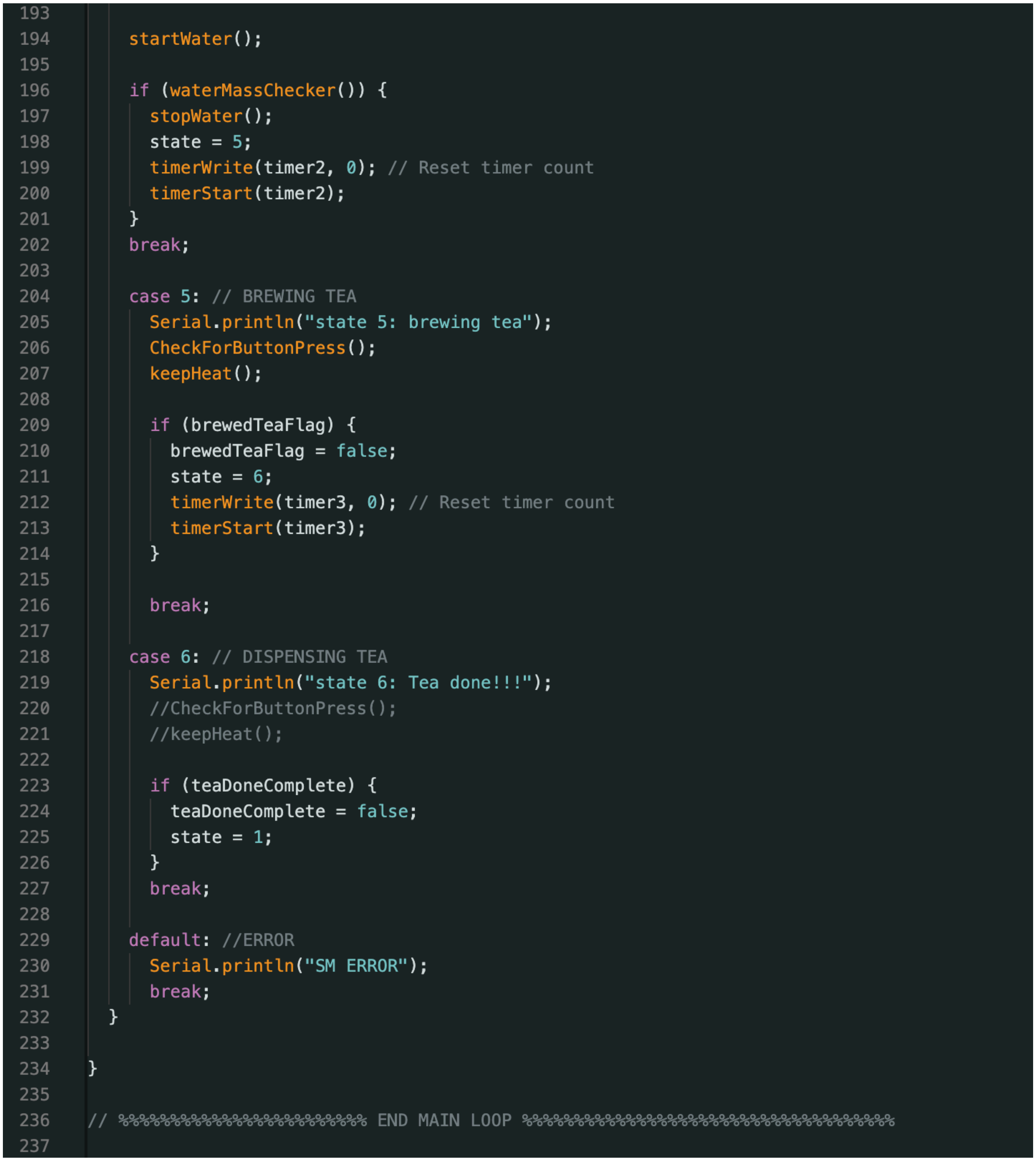
```
float error_sum = 0;

// Motor PWM
const int freq = 5000;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 255;
const int NOM_PWM_VOLTAGE = 150;
volatile int count = 0;                  // encoder count
volatile int count1 = 0;                  // encoder count
volatile int count2 = 0;                  // encoder count
volatile bool deltaT = false;            // check timer interrupt for encoder count
hw_timer_t * timer0 = NULL; // button debouncer
hw_timer_t* timer1 = NULL; // encoder count
hw_timer_t* timer2 = NULL; // brewing tea
hw_timer_t* timer3 = NULL; // dispensing tea
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux2 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux3 = portMUX_INITIALIZER_UNLOCKED;
HardwareSerial mySerial(2);
int button = 10;


// %%% ISRs %%%%
void IRAM_ATTR onTime1() { //encoder speed readoutt
  portENTER_CRITICAL_ISR(&timerMux1);
  count1 = encoder1.getCount();
  encoder1.clearCount();
  count2 = encoder2.getCount();
  encoder2.clearCount();
  deltaT = true;
  portEXIT_CRITICAL_ISR(&timerMux1);
}

void IRAM_ATTR onTime2() {
  portENTER_CRITICAL_ISR(&timerMux2);
  brewedTeaFlag = true; // timer flag for brewing tea
  portEXIT_CRITICAL_ISR(&timerMux2);
  timerStop(timer2);
}

void IRAM_ATTR onTime3() {
  portENTER_CRITICAL_ISR(&timerMux3);
  teaDoneComplete = true; // timer flag for dispensing tea
  portEXIT_CRITICAL_ISR(&timerMux2);
  timerStop(timer3);
}

```



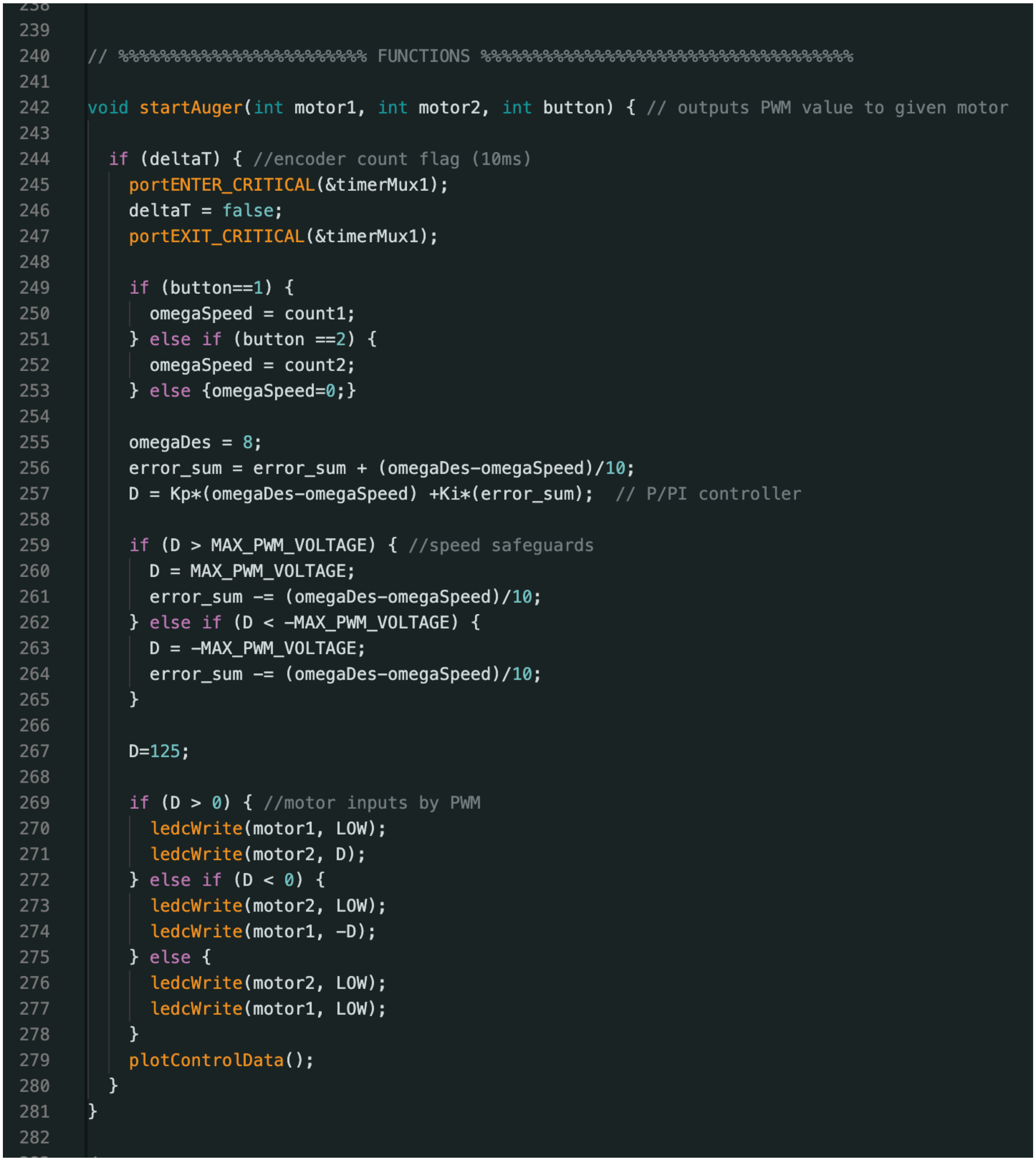
```
void setup() {
  Serial.begin(115200);
  mySerial.begin(9600, SERIAL_8N1, RXD1, TXD1);  // UART setup
  Serial.println("ESP32 UART Receiver");
  ESP32Encoder::useInternalWeakPullResistors = puType::up; //for encoders
  encoder1.attachHalfQuad(34, 39); // Motor 1
  encoder2.attachHalfQuad(36, 4);  // Motor 2
  encoder1.setCount(0);
  encoder2.setCount(0);
  ledcAttach(BIN_1, freq, resolution); //motors
  ledcAttach(BIN_2, freq, resolution);
  ledcAttach(BIN_3, freq, resolution);
  ledcAttach(BIN_4, freq, resolution);
  // pinMode(POT, INPUT);
  pinMode(UPIN, INPUT);
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  pinMode(PUMP, OUTPUT);
  pinMode(HEATER, OUTPUT);
  sensors.begin();

  timer1 = timerBegin(1000000);            // Encoder readout timer: Set timer frequency to
  timerAttachInterrupt(timer1, &onTime1);  // Attach onTimer1 function to our timer.
  timerAlarm(timer1, 10000, true, 0);      // 10000 * 1 us = 10 ms, autoreload true

  timer2 = timerBegin(1000000);            // Tea brewing
  timerAttachInterrupt(timer2, &onTime2);
  timerAlarm(timer2, 1000000*brewTime, true, 0);

  timer3 = timerBegin(1000000);            // Tea dispensing
  timerAttachInterrupt(timer3, &onTime3);
  timerAlarm(timer3, 1000000*dispenseTime, true, 0);
}
```



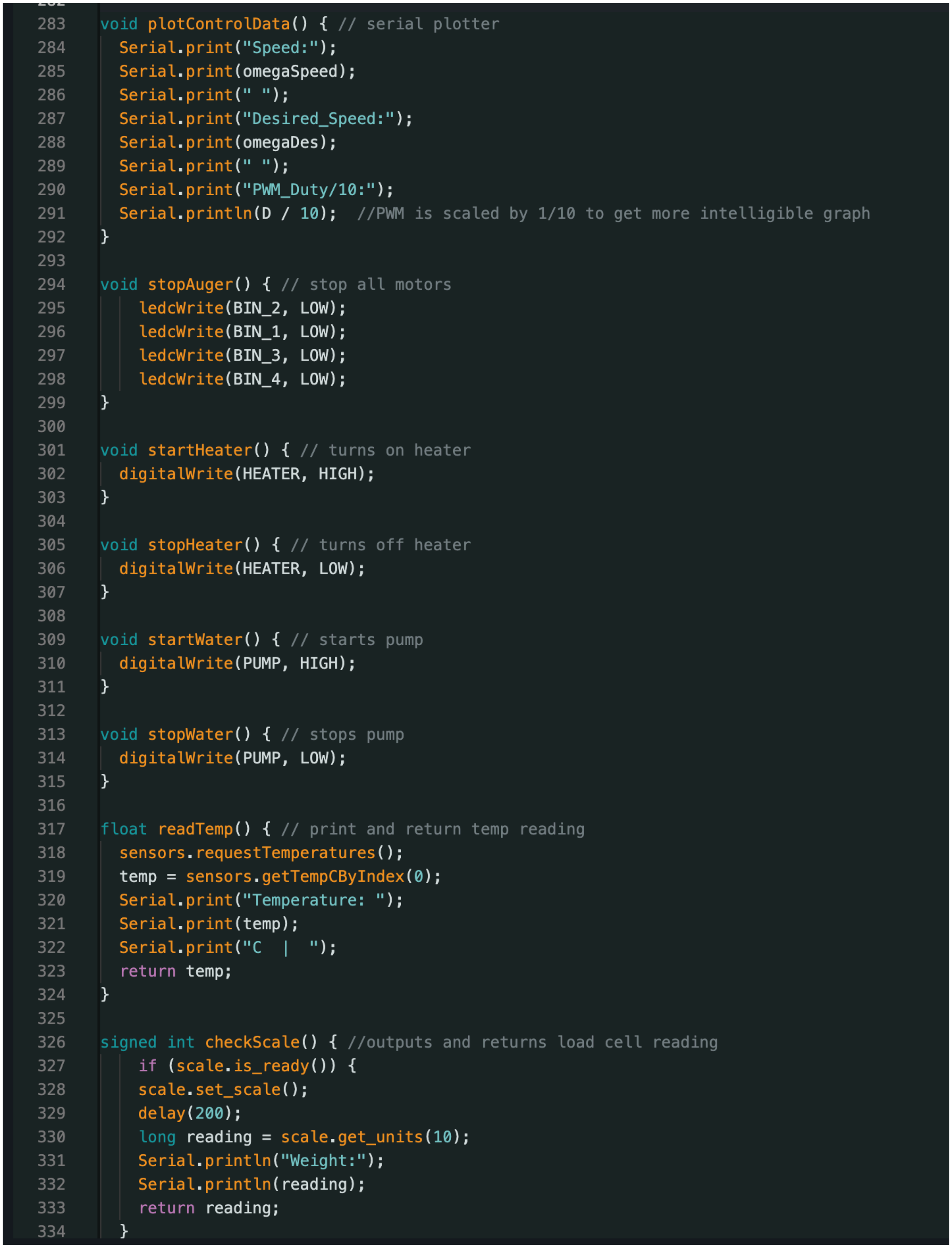
```
void loop() {
  switch (state) {

    case 1: // IDLE
      Serial.println("state 1: idling");
      CheckForButtonPress();
      tareScale();
      checkScale();
      keepHeat();

      if (button == 1) { //tea 1
        Serial.println("button press tea 1");
        motor1 = BIN_1;
        motor2 = BIN_2;
        state = 2;
      }

      else if (button == 2) { //tea 2
        Serial.println("button press tea 2");
        motor1 = BIN_3;
        motor2 = BIN_4;
        state = 2;
      }
      break;

    case 2: // DISPENSING LEAVES
      Serial.println("state 2: dispensing leaves");
      CheckForButtonPress();
      startAuger(motor1, motor2, button);
      //keepHeat();

      if (teaMassChecker()) {
        stopAuger();
        state = 4;
        tareScale();
      }
      break;

    case 3: // PRE-HEATING WATER
      Serial.println("state 3: pre-heating water");
      CheckForButtonPress();
      if (preHeat()) {
        state=1;
      }
      break;

    case 4: // DISPENSING WATER
    Serial.println("state 4: dispensing water");
    stopAuger();
    CheckForButtonPress();
    //keepHeat();
    button = 10;
```



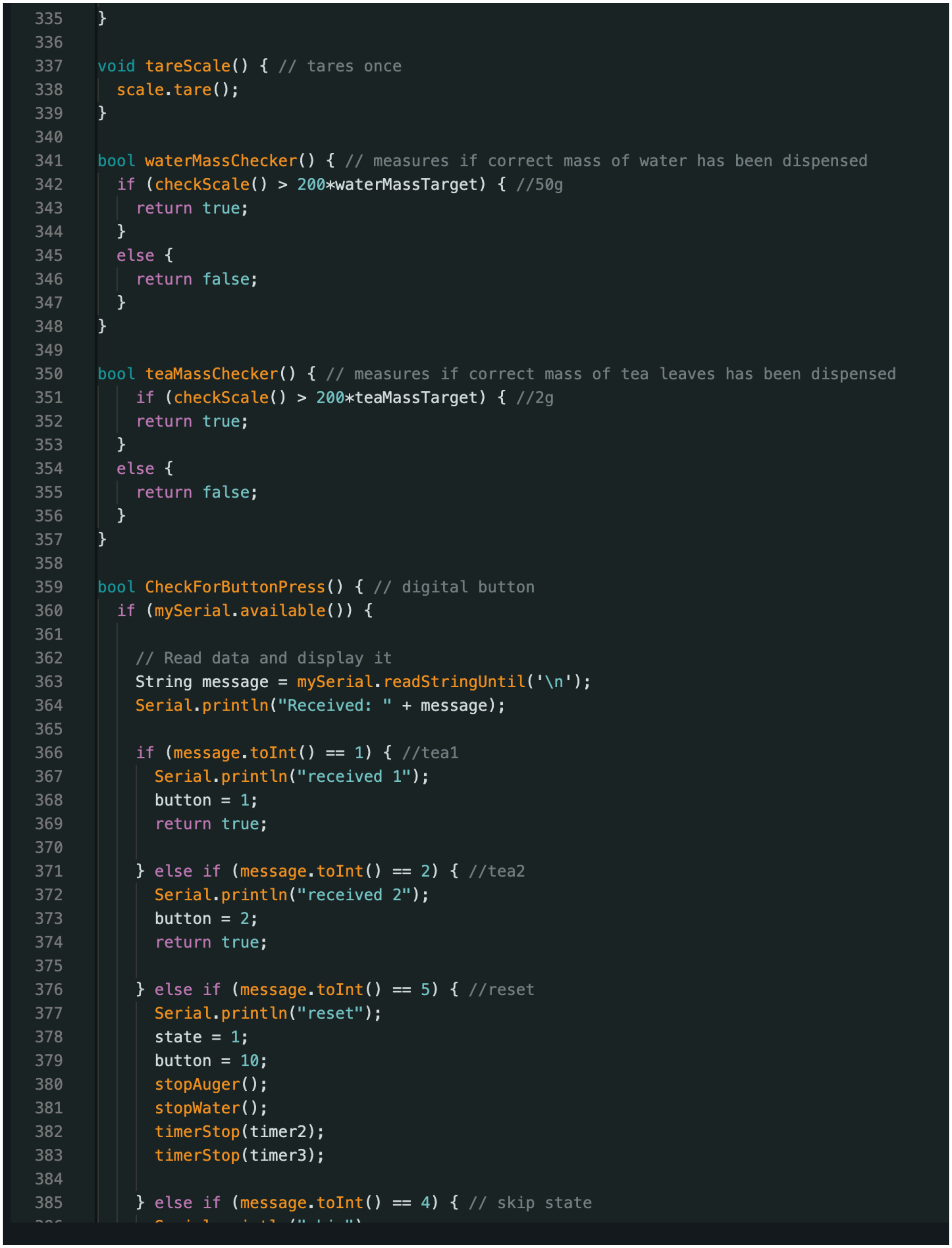
```
    startWater();

    if (waterMassChecker()) {
      stopWater();
      state = 5;
      timerWrite(timer2, 0); // Reset timer count
      timerStart(timer2);
    }
    break;

    case 5: // BREWING TEA
      Serial.println("state 5: brewing tea");
      CheckForButtonPress();
      keepHeat();

      if (brewedTeaFlag) {
        brewedTeaFlag = false;
        state = 6;
        timerWrite(timer3, 0); // Reset timer count
        timerStart(timer3);
      }

      break;

    case 6: // DISPENSING TEA
      Serial.println("state 6: Tea done!!!");
      //CheckForButtonPress();
      //keepHeat();

      if (teaDoneComplete) {
        teaDoneComplete = false;
        state = 1;
      }
      break;

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

}

// %%%%%%%%%%%%%%%%%%%%%%%% END MAIN LOOP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```



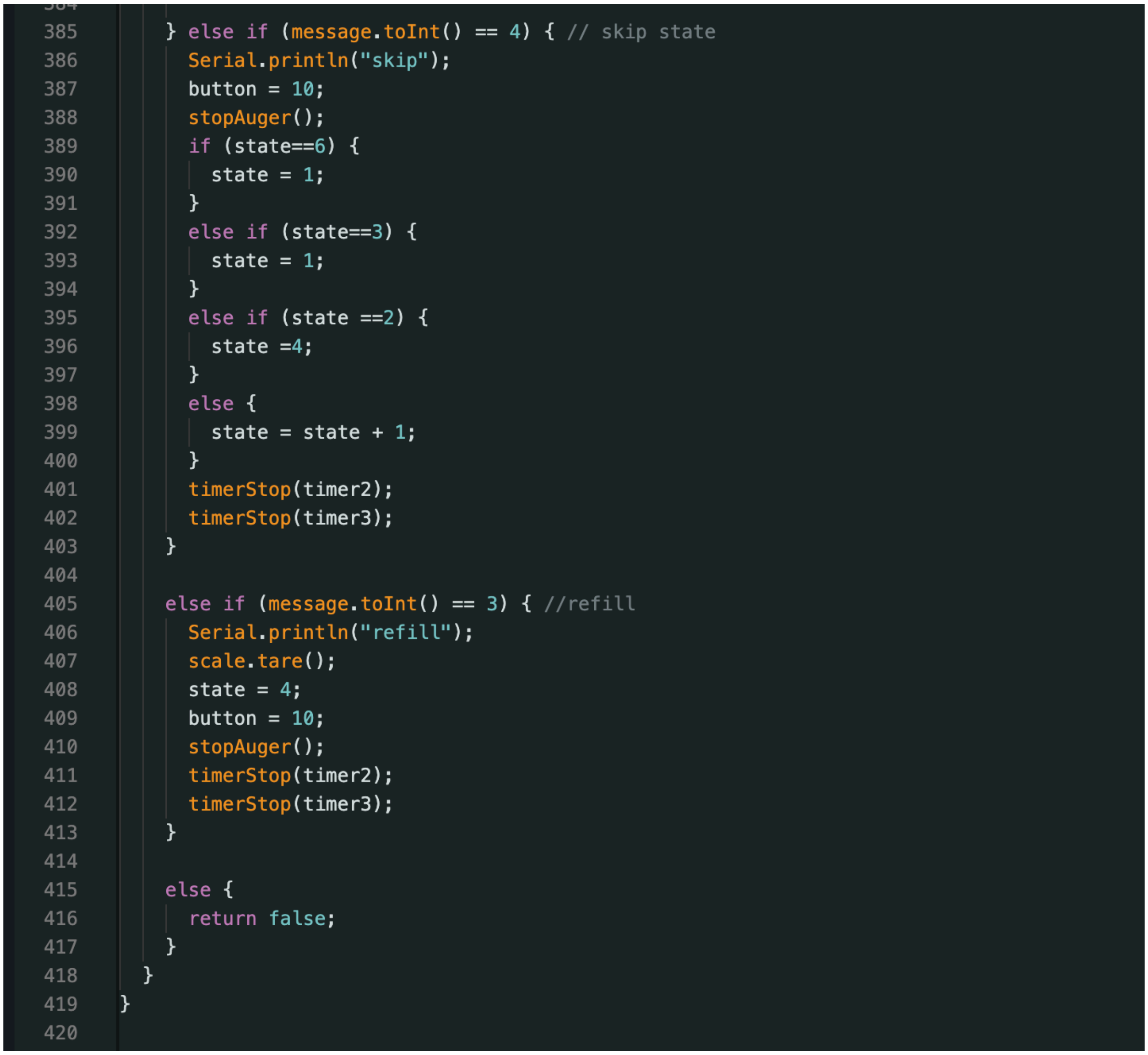
```
// %%%%%%%%%%%%%%%%%%%%%%%% FUNCTIONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

void startAuger(int motor1, int motor2, int button) { // outputs PWM value to given motor

  if (deltaT) { //encoder count flag (10ms)
    portENTER_CRITICAL(&timerMux1);
    deltaT = false;
    portEXIT_CRITICAL(&timerMux1);

    if (button==1) {
      omegaSpeed = count1;
    } else if (button ==2) {
      omegaSpeed = count2;
    } else {omegaSpeed=0;}

    omegaDes = 8;
    error_sum = error_sum + (omegaDes-omegaSpeed)/10;
    D = Kp*(omegaDes-omegaSpeed) +Ki*(error_sum);  // P/PI controller

    if (D > MAX_PWM_VOLTAGE) { //speed safeguards
      D = MAX_PWM_VOLTAGE;
      error_sum -= (omegaDes-omegaSpeed)/10;
    } else if (D < -MAX_PWM_VOLTAGE) {
      D = -MAX_PWM_VOLTAGE;
      error_sum -= (omegaDes-omegaSpeed)/10;
    }

    D=125;

    if (D > 0) { //motor inputs by PWM
      ledcWrite(motor1, LOW);
      ledcWrite(motor2, D);
    } else if (D < 0) {
      ledcWrite(motor2, LOW);
      ledcWrite(motor1, -D);
    } else {
      ledcWrite(motor2, LOW);
      ledcWrite(motor1, LOW);
    }
    plotControlData();
  }
}
```



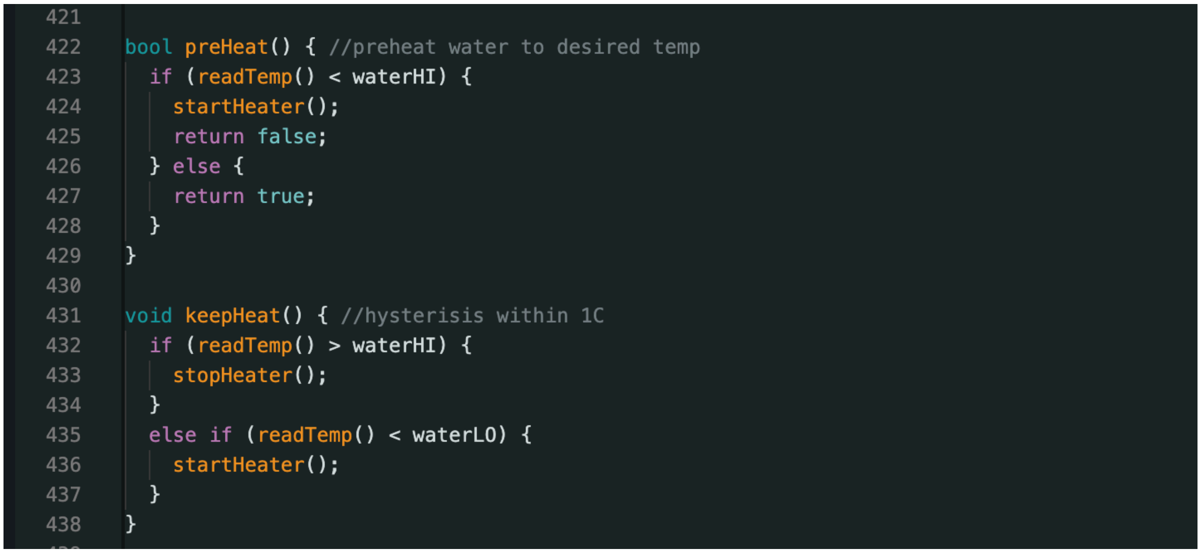
```
void plotControlData() { // serial plotter
  Serial.print("Speed:");
  Serial.print(omegaSpeed);
  Serial.print(" ");
  Serial.print("Desired_Speed:");
  Serial.print(omegaDes);
  Serial.print(" ");
  Serial.print("PWM_Duty/10:");
  Serial.println(D / 10);  //PWM is scaled by 1/10 to get more intelligible graph
}

void stopAuger() { // stop all motors
    ledcWrite(BIN_2, LOW);
    ledcWrite(BIN_1, LOW);
    ledcWrite(BIN_3, LOW);
    ledcWrite(BIN_4, LOW);
}

void startHeater() { // turns on heater
  digitalWrite(HEATER, HIGH);
}

void stopHeater() { // turns off heater
  digitalWrite(HEATER, LOW);
}

void startWater() { // starts pump
  digitalWrite(PUMP, HIGH);
}

void stopWater() { // stops pump
  digitalWrite(PUMP, LOW);
}

float readTemp() { // print and return temp reading
  sensors.requestTemperatures();
  temp = sensors.getTempCByIndex(0);
  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.print("C  |  ");
  return temp;
}

signed int checkScale() { //outputs and returns load cell reading
    if (scale.is_ready()) {
    scale.set_scale();
    delay(200);
    long reading = scale.get_units(10);
    Serial.println("Weight:");
    Serial.println(reading);
    return reading;
  }
```



```
}

void tareScale() { // tares once
  scale.tare();
}

bool waterMassChecker() { // measures if correct mass of water has been dispensed
  if (checkScale() > 200*waterMassTarget) { //50g
    return true;
  }
  else {
    return false;
  }
}

bool teaMassChecker() { // measures if correct mass of tea leaves has been dispensed
    if (checkScale() > 200*teaMassTarget) { //2g
    return true;
  }
  else {
    return false;
  }
}

bool CheckForButtonPress() { // digital button
  if (mySerial.available()) {

    // Read data and display it
    String message = mySerial.readStringUntil('\n');
    Serial.println("Received: " + message);

    if (message.toInt() == 1) { //tea1
      Serial.println("received 1");
      button = 1;
      return true;

    } else if (message.toInt() == 2) { //tea2
      Serial.println("received 2");
      button = 2;
      return true;

    } else if (message.toInt() == 5) { //reset
      Serial.println("reset");
      state = 1;
      button = 10;
      stopAuger();
      stopWater();
      timerStop(timer2);
      timerStop(timer3);

    } else if (message.toInt() == 4) { // skip state
```



```
    } else if (message.toInt() == 4) { // skip state
      Serial.println("skip");
      button = 10;
      stopAuger();
      if (state==6) {
        state = 1;
      }
      else if (state==3) {
        state = 1;
      }
      else if (state ==2) {
        state =4;
      }
      else {
        state = state + 1;
      }
      timerStop(timer2);
      timerStop(timer3);
    }

    else if (message.toInt() == 3) { //refill
      Serial.println("refill");
      scale.tare();
      state = 4;
      button = 10;
      stopAuger();
      timerStop(timer2);
      timerStop(timer3);
    }

    else {
      return false;
    }
  }
}

```



```

bool preHeat() { //preheat water to desired temp
  if (readTemp() < waterHI) {
    startHeater();
    return false;
  } else {
    return true;
  }
}

void keepHeat() { //hysterisis within 1C
  if (readTemp() > waterHI) {
    stopHeater();
  }
  else if (readTemp() < waterLO) {
    startHeater();
  }
}
```