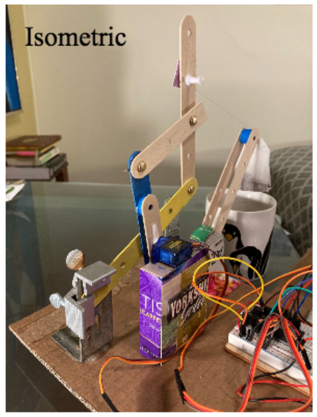
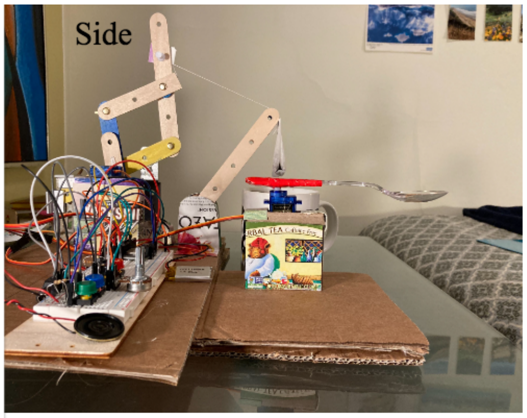
Elsa Robertson
12/10/2020

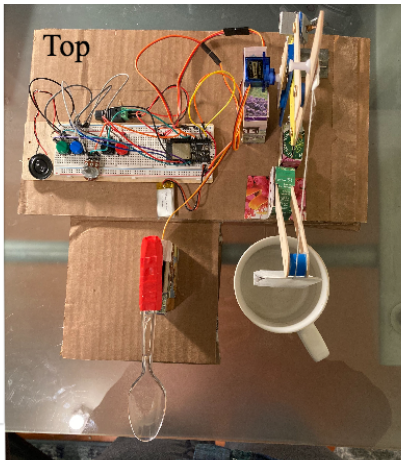
ME-102B Mini-Project: Tea Steeper Prototype

Description of the product:

As a frequent and absentminded tea drinker, I am prone to forgetting when I have begun to boil water and leaving tea to steep far longer than is recommended. This practice can result in a very bitter cup of tea. To solve this problem, I have prototyped a tea steeping device to remove the teabag after a specified period of time. It is mounted on a few sheets of cardboard that sits atop a desk with the mug placed directly on the desk, with the idea that the vertical height could be modified by inserting more cardboard or books if a taller mug is used. It is also limited to the use of tea bags with strings because the string is essential for actuating the intended motion. I found servos to be more suitable for this application than the DC motors we've been using because positional accuracy is important to the functionality while continuous rotation is not. I also wanted to incorporate the mini loudspeaker from the lab kit as a way to communicate with the user without any sort of display.







Electromechanical Details:

Tea Servo: The servo pictured on the above left (which I will now be calling the tea servo) drives a four bar parallel linkage. It is hot glued onto a wooden block which is hot glued onto the cardboard base. The blue popsicle stick serves as the input controlled by the servo and the yellow is a ground. The yellow is fixed by two metal brackets and clamps hot glued to the base. Though in theory this four bar could support a double crank, the motion achieved is that of a double rocker. The output linkage extends beyond the brass fastener pivot and the teabag is fixed to it by stabbing the paper end through a thumbtack (which is hot glued to the popsicle stick). The string of the teabag is draped over a "pulley", which is a plastic tube spacer glued between popsicle sticks, and the string slides against it. With the help of gravity, the two degrees of freedom of the output linkage is reduced to one in which the tea bag is constrained to moving up and down. Through trial and error the desired position range was determined for the tea servo.

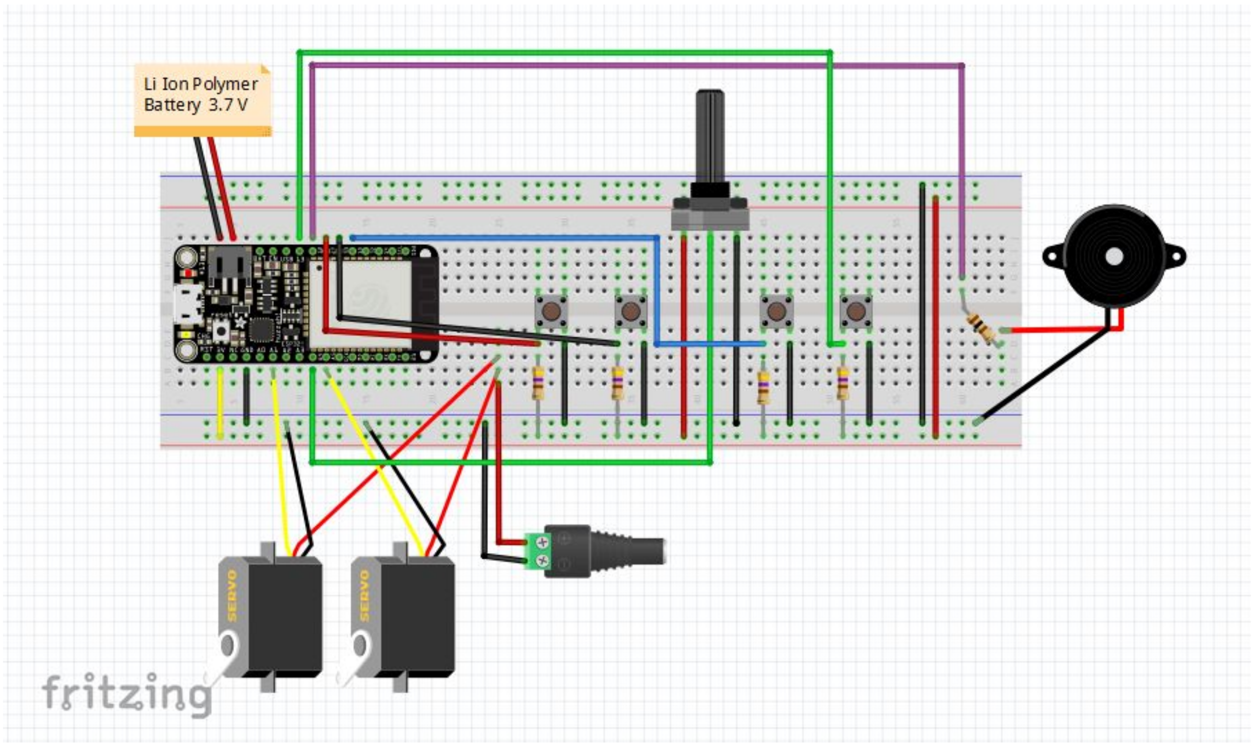
Spoon servo: The motion of the spoon servo, visible in the middle picture above, is considerably simpler. It is also elevated to the desired height on a wooden block, but the spoon servo shaft is glued directly to the spoon so the motion is a simple rotation about the shaft. The limits for position, determined through trial and error, ended up being from position 0 to 95. The motion of the spoon is not the smoothest and likely would have been better if the weight of the spoon was more centered about the axis of rotation.

Circuit:

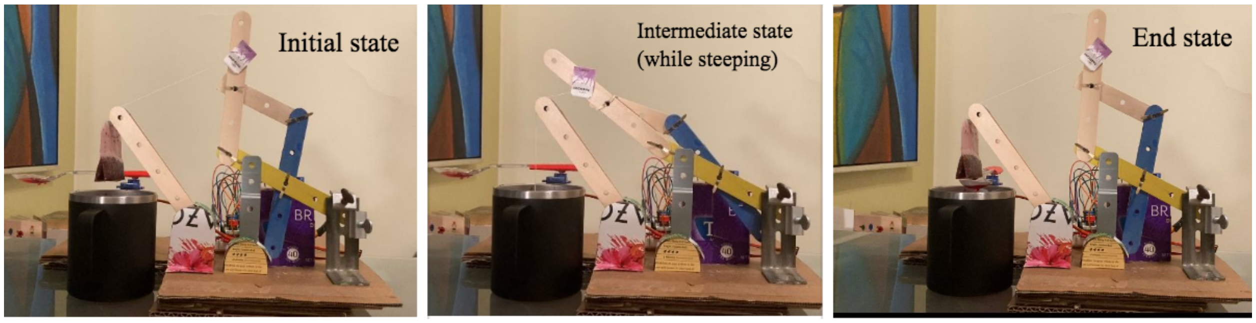
The circuit was an important part of this project because the device requires user input in the form of pressing push-buttons and rotating the potentiometer. The circuit layout itself was fairly straightforward.

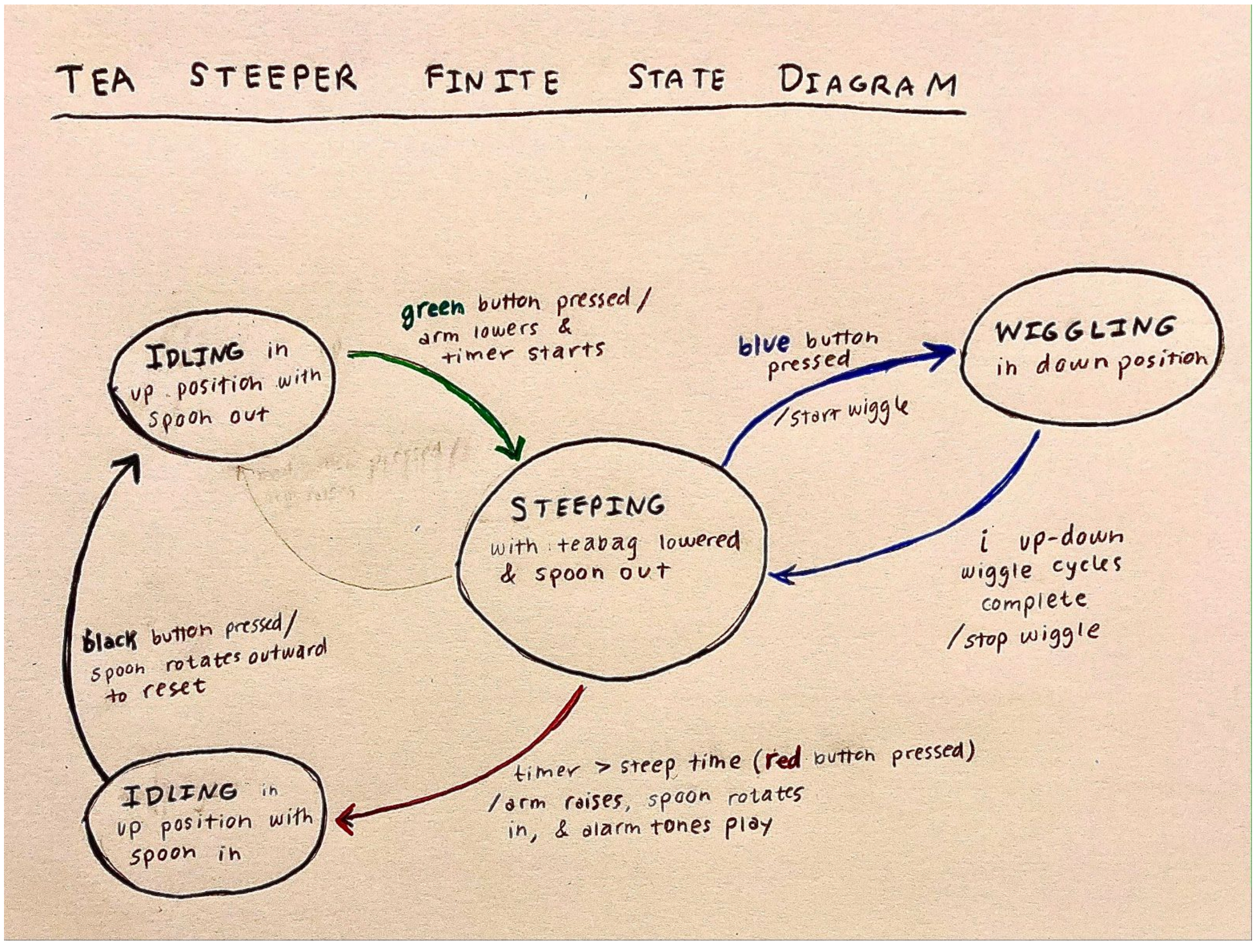
- For the buttons: I tried to use larger buttons and keep them on the near side so that they would be easily accessible during operation. I ordered multicolored Gikfun Momentary Push Buttons (link here:https://www.amazon.com/Gikfun-12x12x7-3-Tactile-Momentary-Arduino/dp/B01E38OS7K/ref=sr_1_4?dchild=1&keywords=gikfun+buttons&qid=1607667225&sr=8-4) so that each color would be associated with a different function. Each is set up as an input to a GPIO pins with approximately 470 ohm (didn't have enough of the same type) as a pull-up resistor so that the signal is held high whenever buttons are not pressed. The 3.7V Li Ion Polymer Battery is plugged into the ESP32 to power it and the 3V pin on the micro is connected to the power rails of the breadboard to power the components (I may have misspoke in the video). When a button is pressed it triggers its corresponding interrupt, causing events in event-checkers to be carried out and the associated services. More details on this in the finite state diagram coming soon. A debounce is also included for each to prevent misfiring.
- For the potentiometer: I used the potentiometer out of the kit connected to the power rail, common ground, and as an input to an analog pin. The readings from this were mapped to pitch frequencies to be used by the speaker and also to steep time states. For steep time the range of the potentiometer reading was split into 5 sections and each section was discretized into a steep time of 2, 3, 4, 5, or 6 minutes (in ms in arduino). When a steep time threshold is crossed, a pin output is sent to the mini loudspeaker to play a beep of the mapped frequency. This is a way of notifying the user what the time is set to. This could still be improved by adding a hysteresis band.
- For the mini loudspeaker: This is also from the kit. I found online that it is best with a 100 ohm resistor in series with an output pin (and also connected to the ground rail). For ESP32 it is configured using PWM. I initially had the channel set to 0 which set me back significantly as it interfered with the servos (whose PWM channels aren't explicitly defined in my code but still use them) but once I changed it to 5 could be adjusted in volume and pitch with various ledcWrite commands. I ended up using delays to set tone duration, which is not ideal for overall code efficiency but is simpler to implement and worked okay for my purposes.

- For the servos: I used 2 SG90 9g Micro Servos (4 for ~$9 on amazon). I powered these directly from the 5V power source, as seen in the Fritzing diagram below, and they are also connected to analog output pins. Using the ServoESP32 library by Jaroslav Paral, each servo could be told to go to the desired position in degrees.



Finite state machine:
This device has four states and transitions are triggered by button presses or timers. The initial state is to be idling with the teabag in the “up” position and the spoon in the “out” position. When the green button is pressed, the steeping begins by lowering the teabag and starting the timer, completing the transition to the “steeping” intermediate state in the diagram. From here the most probable mode of use is to wait for the time to run out. When the timer surpasses the set steep time, the tea arm raises, the spoon is rotated “in” beneath the raised teabag to prevent drippage, and an alarm tone sequence plays. Alternatively, the user has the option to cut the steep time short and jumpstart the transition to the end state by pressing the red button, which triggers the same operation as above aside from a slightly different tone. From the end state, the user can reset to the initial state by pressing the black button which moves the spoon from the “in” to the “out” position (though it is recommended to remove and dispose of the used teabag first). Additionally, there is an optional fourth state (check out video for visual) that can be achieved by pressing the blue button. This introduces a temporary wiggle state in which the tea arm already in the “down” position bounces a few degrees up and down from this point to promote mixing. The machine is set to automatically return to the intermediate state once 10 up-down cycles are completed.


Initial state
Intermediate state (while steeping)
End state


TEA STEEPER FINITE STATE DIAGRAM
IDLING in up position with spoon out
green button pressed / arm lowers & timer starts
STEEPING with teabag lowered & spoon out
blue button pressed / start wiggle
WIGGLING in down position
i up-down wiggle cycles complete / stop wiggle
timer > steep time (red button pressed) / arm raises, spoon rotates in, & alarm tones play
IDLING in up position with spoon in
black button pressed / spoon rotates outward to reset

Appendix: Arduino Code

[code]

```
#include <Servo.h>  //downloaded ServoESP32 library by Jaroslav Paral
Servo myservo; //create servo object for spoon
Servo myservo1;//create servo object for tea lift

// inputs
#define greenPin 13
#define blackPin 15
#define bluePin 14
#define redPin 33
#define potPin A4

//outputs
#define speakerPin 12
#define servopin  4
#define servopin1  25

//PWM for speaker
#define channel 5 //must be different than pwm channels used by servos
#define resolution 8 //duty cycle must be betweeen 0 and 255
#define freq 2000

//misc variables
#define DEBOUNCE 200
int pos = 0; // position of spoon servo
int pos1 = 0; // position of lift servo
int down = 60; //final position when lift is down
int up = 10; //final position when lift is up
int in = 95; //final position when spoon is in
int out = 0; //final posiiton when spoon is out
int potReading; //reading on pot from 0-4095
int speakerMap; //map pot reading to audio frequencies
int startsteep = 0;
int steeptime = 0;
int timesteeped = 0;
int count = 0;
int i=0;

//button states and timers
```

```
volatile boolean greenPressEvent = false;
volatile int greenTimer = 0;
volatile int greenTimer_last = 0;
volatile boolean bluePressEvent = false;
volatile int blueTimer = 0;
volatile int blueTimer_last = 0;
volatile boolean blackPressEvent = false;
volatile int blackTimer = 0;
volatile int blackTimer_last = 0;
volatile boolean redPressEvent = false;
volatile int redTimer = 0;
volatile int redTimer_last = 0;

//potentiometer states
volatile int statePot = 0; //current state (to determine steep time) based on pot thresholds
volatile int statepotold = 0;//previous state
volatile boolean stateEvent = false; //turns to true whenever a button press event has occurred.
volatile int stateTimer = 0;
volatile int stateTimer_last = 0;

void setup() {
  pinMode(greenPin, INPUT);
  pinMode(bluePin, INPUT);
  pinMode(blackPin, INPUT);
  pinMode(redPin, INPUT);
  pinMode(servopin, OUTPUT);
  pinMode(servopin1, OUTPUT);
  pinMode(speakerPin, OUTPUT);
  myservo.attach(servopin);  // attaches the servo on pin 4/ A5 to the servo object
  myservo1.attach(servopin1);

  attachInterrupt(digitalPinToInterrupt(greenPin), greenIsPressed, RISING);
  attachInterrupt(digitalPinToInterrupt(bluePin), blueIsPressed, RISING);
  attachInterrupt(digitalPinToInterrupt(blackPin), blackIsPressed, RISING);
  attachInterrupt(digitalPinToInterrupt(redPin), redIsPressed, RISING);

  ledcSetup(channel, freq, resolution); //channel, freq, resolution
  ledcAttachPin(speakerPin, channel);
  Serial.begin(115200);
}
```

```
void loop() {
  //assign state of steep time based on potentiometer reading
  //would be better to add hysteresis
  potReading = analogRead(potPin);
  speakerMap = map(potReading, 0, 4095, 255, 10000);
  if (potReading < 819) {
    statePot = 0;
    steeptime = 120000;//2 min
  }
  if ((potReading < 1638) && (potReading >= 819)) {
    statePot = 1;
    steeptime = 180000; //3 min
  }
  if ((potReading < 2457) && (potReading >= 1638)) {
    statePot = 2;
    steeptime = 240000; //4 min
  }
  if ((potReading < 3276) && (potReading >= 2457)) {
    statePot = 3;
    steeptime = 300000; //5 min
  }
  if (potReading >= 3276) {
    statePot = 4;
    steeptime = 360000; //6 min
  }

  // EVENT CHECKERS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  stateTimer = millis(); //check if steep time has changed
  if ((statepotold != statePot) && (stateTimer - stateTimer_last > DEBOUNCE)) {
    Serial.print(statePot);
    Service4(); //beep when threshold is crossed to communicate state to user
    stateTimer_last = stateTimer;
    statepotold = statePot;
  }

  if ( bluePressEvent == true) { // wiggle teabag up and down for aggressively diffused tea
    Service5(); //Service routine
    bluePressEvent = false; //reset event flag
  }
  if ( greenPressEvent == true) {//start steeping tea
    Service0(); //lower arm
    startsteep = millis();//start timer
    count = 1;
    greenPressEvent = false; //reset event flag
  }
  if ( blackPressEvent == true) {//reset spoon
    Service3(); // swing spoon arm out
    blackPressEvent = false; //reset event flag
  }
  if ( redPressEvent == true) {//exit early from steeping for impatient users
    Service4();//beep three times
    Service4();
    Service4();
    Service1(); // raise teabag arm
    Service2(); // insert spoon to catch teabag drips
    count = 2; //add to count so that it doesn't try to move at the scheduled time
    redPressEvent = false; //reset event flag
  }
  timesteeped = millis() - startsteep;
  if ((timesteeped > steeptime) && (count == 1)) { //check if tea has been steeped long enough
    Service1(); //raise teabag
    Service2();//insert spoon to catch drips
    Service6();//play alarm
    count = 2; // use count to make it can only be done once
  }
}
//SERVICE ROUTINES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
void Service0() {
  //lower arm
  for (pos1 = up; pos1 <= down; pos1 += 1) {
    // in steps of 1 degree
    myservo1.write(pos1);              // tell servo to go to position in variable 'pos'
    delay(20);                       // waits 15ms for the servo to reach the position
  }
}
void Service1() {
  //raise arm
  for (pos1 = down; pos1 >= up; pos1 -= 1) {
    // in steps of 1 degree
    myservo1.write(pos1);              // tell servo to go to position in variable 'pos'
    delay(20);                       // waits 20ms for the servo to reach the position
  }
}
void Service2() {
  //insert spoon
  for (pos = out; pos <= in; pos += 1) {
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  /*for (pos = in; pos >= out; pos -= 1) {
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(20);                       // waits 15ms for the servo to reach the position
    }/*/
}
void Service3() {
  //remove spoon
  for (pos = in; pos >= out; pos -= 1) {
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
void Service4() { //beep when steep time is changed
  ledcWrite(channel, 125);
  ledcWriteTone(channel, speakerMap);
  delay(100);
  ledcWriteTone(channel, 0);
  delay(100);
}
void Service5() {
  for (i = 1; i <= 10; i += 1) {
    //lower arm
    for (pos1 =45; pos1 <= 60; pos1 += 1) { // goes from 45 degrees to 60 degrees
      // in steps of 1 degree
      myservo1.write(pos1);              // tell servo to go to position in variable 'pos'
      delay(20);                       // waits 15ms for the servo to reach the position
    }
    //raise arm
    for (pos1 = 60; pos1 >= 45; pos1 -= 1) { // goes from 60 degrees to 45 degrees
      // in steps of 1 degree
      myservo1.write(pos1);              // tell servo to go to position in variable 'pos'
      delay(20);                       // waits 20ms for the servo to reach the position
    }
  }
}
void Service6() { //beep when steep time is changed
  ledcWrite(channel, 125);
  ledcWriteTone(channel, 131);
  delay(300);
  ledcWriteTone(channel, 175);
  delay(350);
  ledcWriteTone(channel, 330);
  delay(400);
  ledcWriteTone(channel, 523);
  delay(450);
  ledcWriteTone(channel, 1047);
  delay(500);
  ledcWriteTone(channel, 0);
}
//INTERRUPT SERVICE RESPONSES %%%%%%%%%%%%%%%%%%%%%%
void greenIsPressed() {
  greenTimer = millis();
  Serial.print("green! ");
  if (greenTimer - greenTimer_last > DEBOUNCE) {
    greenPressEvent = true;
    greenTimer_last = greenTimer;
  }
}
void blueIsPressed() {
  blueTimer = millis();
  if (blueTimer - blueTimer_last > DEBOUNCE) {
    bluePressEvent = true;
    blueTimer_last = blueTimer;
  }
}
void blackIsPressed() {
  blackTimer = millis();
  Serial.print("black! ");
  if (blackTimer - blackTimer_last > DEBOUNCE) {
    blackPressEvent = true;
    blackTimer_last = blackTimer;
  }
}
void redIsPressed() {
  redTimer = millis();
  if (redTimer - redTimer_last > DEBOUNCE) {
    redPressEvent = true;
    redTimer_last = redTimer;
  }
}
```

[/code]