The Egg Smasher

By Shane Pauker

Description of the Product:

Waste is a major concern in modern food systems, as it unnecessarily expends money and resources. Companies such as Ugly Produce are trying to reduce food waste by selling traditionally unmarketable (but perfectly edible) produce; organizations such as Cal Dining are trying to reduce waste in their services. As a researcher in the plant-based food space, food waste is of paramount importance to me. Despite these concerns, however, I simply thought it would be funny to make an artistic device that smashes an egg and makes a mess on your countertop. It serves no other purpose. It doesn't even smash the egg in a clean way! It usually breaks the yolk and gets flecks of shell into the egg.

Electromechanical details:

Interfacing with the Egg: The egg to be broken is inserted into a small pouch, which hangs from a counterweighted seesaw. A large egg (by USDA standards) is just heavy enough to tip this seesaw. The change in pressure is detected by a Force-Sensitive Resistor (FSR), which communicates with the microcontroller to trigger an egg-breaking arm.

The seesaw is somewhat rudimentary, as precision and load transfer were not of particular importance in its function. Seesaw uses a ⅜" bolt as the primary axle, fixed and located by nuts and washers. There is no bearing in the rotating seesaw body; allowing the friction of the bolt against wood was acceptable for these purposes. Furthermore, the mechanism is made entirely of parts that I already had in my house: scrap wood and spare hardware. For a higher-budget project, a bearing-based assembly would likely sustain a longer life cycle.

Figure 1: The Egg Smasher, unloaded (left) and loaded (right). In the loaded case, the motor is unplugged, such that the photo could be taken.

Breaking the Egg: Once the appropriate pressure scenario is detected, the system waits momentarily; this allows the user to move their hand out of the way and for the egg to fully drop

to the bottom of the system. After waiting, a motor sitting atop the seesaw swings an arm, hitting the egg. The arm pauses again, briefly, before returning to the original position.

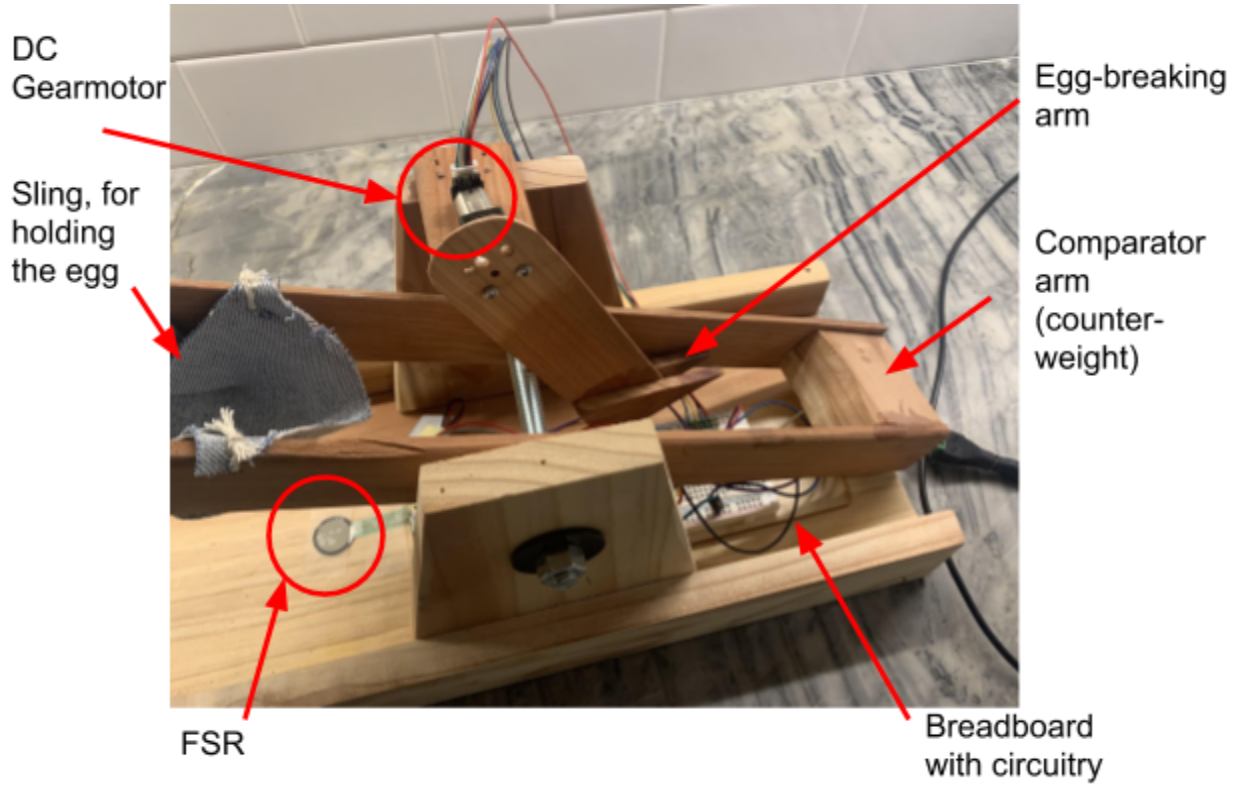

Figure 2: The egg, loading into the swing, lowers the comparator arm, pressing on the FSR. Per the code, the egg-breaking arm waits briefly before breaking the egg.

Circuit:
Beyond the microcontroller (ESP32 Feather), there were two main components used:

1. The DC Gearmotor: This was the provided DC gearmotor. It was driven using the gearmotor provided in our class lab kit(Pololu Part 2215). It was driven using the DRV8833 Dual Motor Driver Carrier, also provided in our lab kits, and powered with a 9V barrel jack that I typically use to power guitar pedals.
2. The FSR: The FSR was Pololu Part 1696 (~$10.85), which is a 0.6” diameter circular FSR. It is a resistor whose resistance decreases nonlinearly when a force is applied. It was powered by the microcontroller, an appropriate resistor was used, and the signal running through it was treated as analog for pressure detection. This was the only part I purchased specifically for this project.

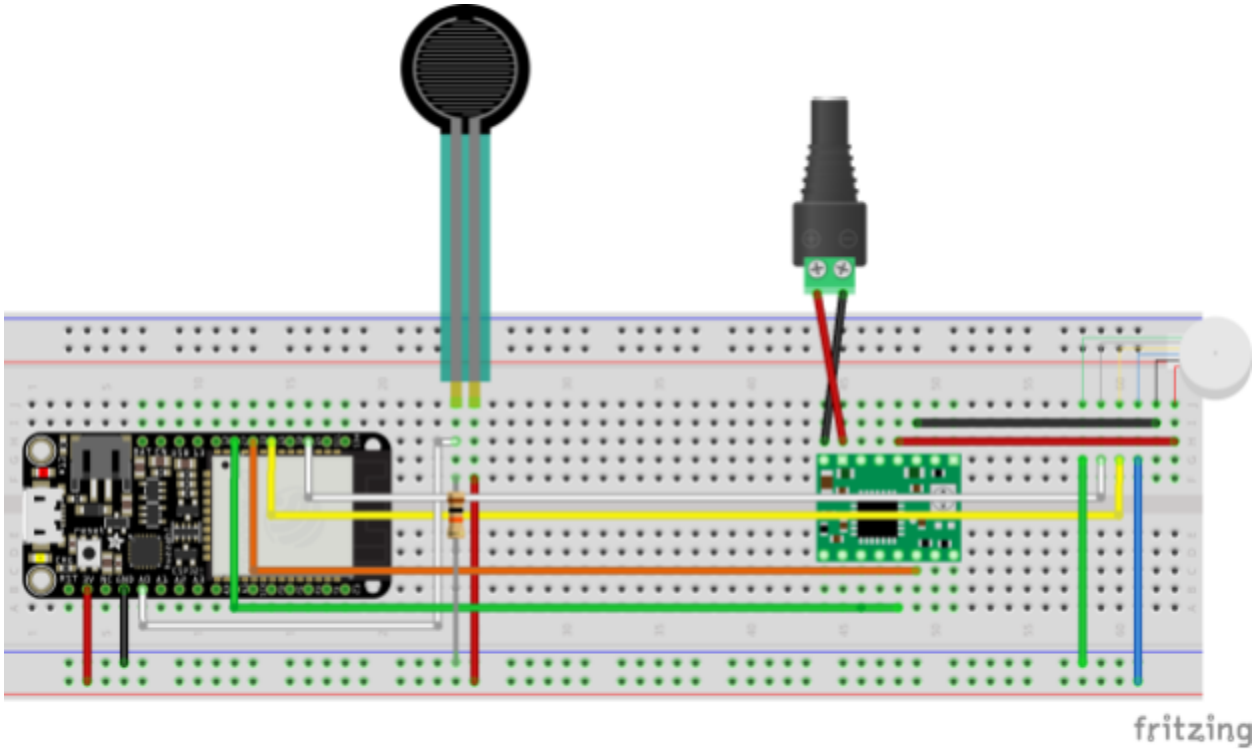

Figure 3: Circuit diagram for this machine

Finite State Machine:
The finite state machine for this project is essentially a linear progression through a series of events. By default, the system is idling. When the FSR is triggered, it begins a timer. When the timer is finished, the smashing arm swings into place, smashing the egg. Finally, the arm resets itself.

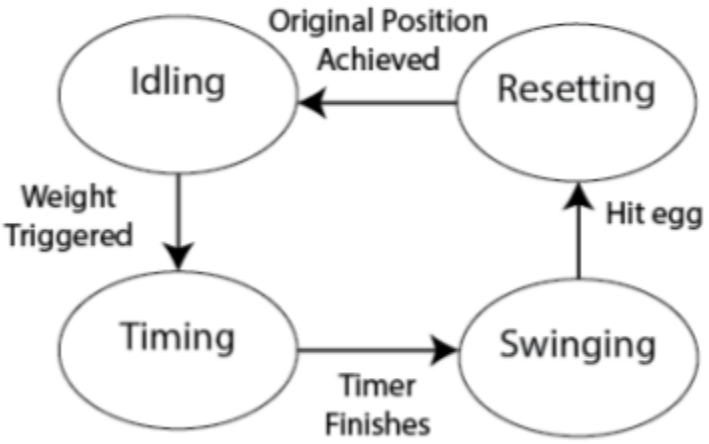

Figure 4: Finite-state diagram for this machine. Idling checks for FSR pressure, timing waits briefly, swinging causes the egg-breaking arm to break the egg, and resetting returns the arm to its initial position.

It should be noted that in the final implementation of the code, the swing arm position is determined through a series of timers rather than true position. This was a last-minute workaround, as the motor encoder ceased communicating with the MCU near the end of my testing and soon before I began shooting video.

Appendix I: Arduino Code

```
// Shane Pauker
// ME 102B Mini-Project
// Implement open-loop velocity/pwm control for our motor

//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
// INCLUDE libraries
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

#include <Arduino.h>
portMUX_TYPE timerMUX = portMUX_INITIALIZER_UNLOCKED;

#include <ESP32Encoder.h>
ESP32Encoder encoder;
ESP32Encoder encoder2;

//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
// DEFINE constants
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

// Pins
#define builtIn 13   //  built-in LED pin
#define motPin1 27   //  motor GPIO pin  (the 6th one)
#define motPin2 33   //  motor GPIO pin  (the 7th one
#define encPinA 15   //  encoder Pin A   (the 8th one)
#define encPinB 14   //  encoder Pin B   (the 10th one)
uint8_t pressurePin = A0; //  pressure sensor pin (the 9th one)


// Channels
#define channel1 1 //  channel for motor
#define channel2 2 //  channel for motor

// Timer
hw_timer_t * timer = NULL; // for switch interrupts
```

```
int timerLength = 100000; // for setting up timer
int waitLength = 1500000; // for setting up timer
int swingLength = 500000; // for swing duration

// Motor rotation
#define myFreq 5000 // frequency for ledcsetup
#define myRes 8     // PWM resolution
#define myRot 200   // rotation angle
#define presThresh 1500 // threshold for egg trigger
float myCount = 0;

//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
// DEFINE Variables
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
int pressureVal = 0;
int state = 0;
int forwardSwing = 0;
int swingBack = 0;
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
// DEFINE interrupts
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

void onTimer() {
  // Compute Velocity
  myCount = float(encoder.getCount());
  Serial.println(myCount);
  Serial.println(state);
}

void onWait() {
  Serial.println("3 seconds has elapsed");
  state = 2;
  Service2();
}
```

```
void onSwing() {
  forwardSwing = 1;
}

void backSwing() {
  swingBack = 1;
}

//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
// DEFINE initialization
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

void setup() {
  // Configure Pins
  pinMode(builtIn, OUTPUT); //  built-in LED pin
  pinMode(motPin1, OUTPUT); // Motor Pin 1
  pinMode(motPin2, OUTPUT); // Motor Pin 2
  pinMode(pressurePin, INPUT); // button pin


  // Set up encoder
  ESP32Encoder::useInternalWeakPullResistors = UP;   // Enable the weak
pull up resistors
  encoder.attachFullQuad(encPinA, encPinB);  // Attach pins for use as
encoder pins
  encoder.clearCount();   // set starting count value after attaching

  // Set up + attach channels
  ledcSetup(channel1, myFreq, myRes);
  ledcSetup(channel2, myFreq, myRes);
  ledcAttachPin(motPin1, channel1);
  ledcAttachPin(motPin2, channel2);

  // Start motor coasting
  ledcWrite(channel1, 0);
  ledcWrite(channel2, 0);

  // Start Timer
  timer = timerBegin(0, 80, true); // timer_id = 0; divider=80; countUp =
```

```
true;
  timerAttachInterrupt(timer, &onTimer, true); // edge = true
  timerAlarmWrite(timer, timerLength, true);  //1000 ms
  timerAlarmEnable(timer);

  // Initiate serial
  Serial.begin(115200);
}

/////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
// DEFINE main loop
/////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

void loop() {
  switch (state) {
    case 0: // No motor motion, only checking pressure pad
      pressureVal = analogRead(pressurePin);
      if (pressureVal > presThresh) {
        state = 1;
        Service1();
      }
      break;
    case 1: // Waits for the predetermined period
      break;
    case 2:
      if (forwardSwing == 1) {
        Service3();
        state = 3;
        forwardSwing = 0;
      }
      break;
    case 3:
      if (swingBack == 1) {
        Service0();
        state = 0;
        swingBack = 0;
      }
      break;
  }
```

```
}
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
// DEFINE services
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
void Service1() {
  // Start Timer
  timer = timerBegin(0, 80, true); // timer_id = 0; divider=80; countUp =
true;
  timerAttachInterrupt(timer, &onWait, true); // edge = true
  timerAlarmWrite(timer, waitLength, true);  //1000 ms
  timerAlarmEnable(timer);
}

void Service2() {
  timer = timerBegin(0, 80, true); // timer_id = 0; divider=80; countUp =
true;
  timerAttachInterrupt(timer, &onSwing, true); // edge = true
  timerAlarmWrite(timer, swingLength, true);
  timerAlarmEnable(timer);
  // Turns motor on
  ledcWrite(channel1, 255);
  ledcWrite(channel2, 0);
}

void Service3() {
  timer = timerBegin(0, 80, true); // timer_id = 0; divider=80; countUp =
true;
  timerAttachInterrupt(timer, &backSwing, true); // edge = true
  timerAlarmWrite(timer, swingLength, true);
  timerAlarmEnable(timer);
  // Turns motor on (in reverse!)
  ledcWrite(channel1, 0);
  ledcWrite(channel2, 255);
}

void Service0() {
  originalTimer();
  // Turns motor off
  ledcWrite(channel1, 0);
```

```
  ledcWrite(channel2, 0);
}

void originalTimer() {
  // Start Timer
  timer = timerBegin(0, 80, true); // timer_id = 0; divider=80; countUp =
true;
  timerAttachInterrupt(timer, &onTimer, true); // edge = true
  timerAlarmWrite(timer, timerLength, true);  //1000 ms
  timerAlarmEnable(timer);
}
```