ME 102B Final Project Report: Pet Interaction Extravaganza (P.I.E.)

Keilani Adachi, Sydney Chan, Laura Dabundo, and Nate Pellini

Opportunity: How might we interact with and care for our pets with minimal effort?

High Level Strategy and Functionality: Create a button-operated automated feeder to dispense a predetermined amount of dry food. Create a ball launcher which a dog can operate by dropping a ball into a tube and pawing a sensor. Initially, our intentions were to also have a treat dispenser, feather toy, and owner-pet interactions through video and audio communication. However, it quickly became apparent that these functions were out of the feasible scope of this project. The feeder was originally designed to be scheduled, but it was more practical to make it button activated.

Integrated Physical Device:

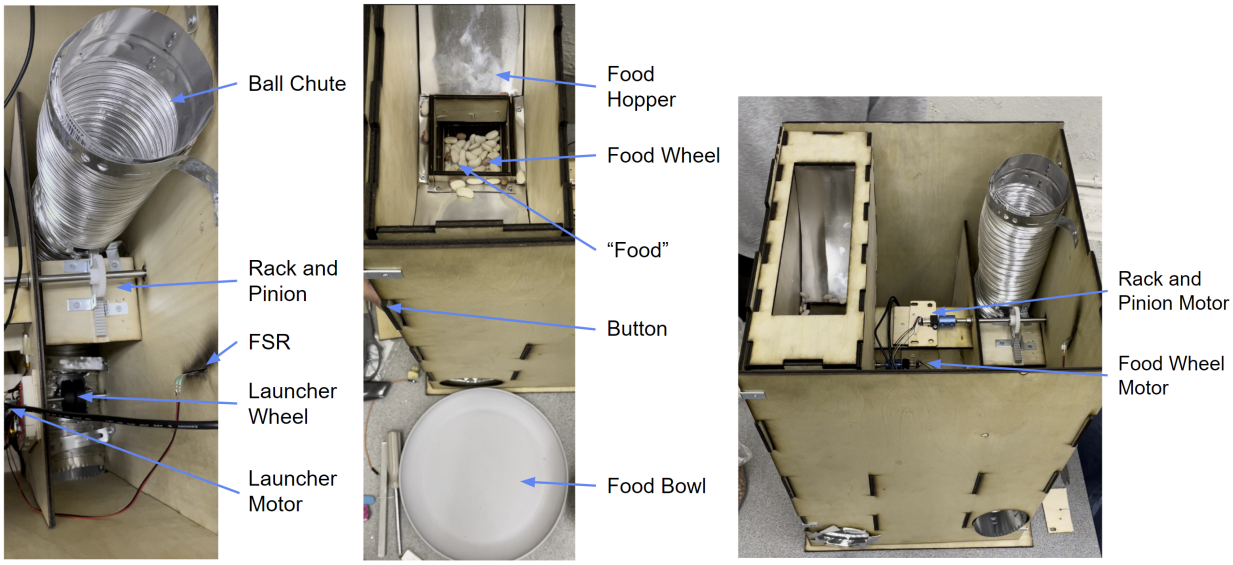


Figure 1. (a) Ball Launcher Assembly including a rack and pinion, a launcher wheel, and a ball chute. (b) Food Wheel Assembly including a food storage hopper, food wheel, beans as food, and a food bowl. (c) The full P.I.E. assembly with the ball launcher on the right and the feeder on the left.

Decisions and Calculations: We chose bearings and motors that would be able to support the following calculations:

[Ball Launcher Bearings]

$$m_{wiffle\ ball} \approx 0.05\ kg\ ,\ m_{lego\ wheel} \approx 0.02\ kg$$

$$F_{g,\,total} \approx (0.05 + 0.02)\ kg\ *\ (9.8\ \frac{m}{s^2}) = 0.686\ N$$

$$with\ 2\ bearings\ ,\ F_g = 0.343\ N\ per\ bearing$$

[Ball Launcher Motor]

$$I_{wheel} = \frac{1}{2}m(r_o^2 - r_i^2) = \frac{1}{2}(.02\ kg)(0.127^2 - 0.001^2) = 1.6 \times 10^{-6} kg \bullet m^2)$$

$$speed\ from\ 0\ to\ full\ velocity\ in\ t = 0.25\ s$$

$$ball\ launch\ speed:\ v = 4\frac{m}{s},\ \omega = \frac{v}{r} = \frac{4}{0.0127} \approx 315\ \frac{rad}{s},\ \alpha = \frac{\Delta\omega}{\Delta t} = \frac{315}{0.25} \approx 1260\ \frac{rad}{s^2}$$

$$\tau = I\alpha = 2.016 \times 10^{-3} N \bullet m$$

[Food Wheel Bearings]

$$V_{kibble} = 220\ in^3 \rightarrow m_{kibble} = 15\ cups * \frac{4\ oz\ kibble}{1\ cup} \approx 0.05\ kg \rightarrow F_{kibble} \approx 17N$$

$$V_{food\ wheel} = 0.001304\ m^3 \rightarrow m_{food\ wheel} \approx V_{food\ wheel} * 500\ \frac{kg}{m^3} \rightarrow F_{food\ wheel} \approx 6.39N$$

$$F_{g,\,total} = 17N + 6.39N \approx 24\ N$$

$$with\ 2\ bearings,\ F_g = 12\ N\ per\ bearing$$

[Food Wheel Motor]

$$say\ \omega = \pi/2\ \frac{rad}{s},\ then\ to\ accelerate\ in\ 1s,\ \alpha = \frac{\Delta\omega}{\Delta t} = \pi/2\ \frac{rad}{s^2}$$

$$I_{cylinder} = \frac{1}{2}mr^2 = \frac{1}{2}(.652\ kg)(0.075m)^2 = 0.0018kg \bullet m^2)$$

$$\tau = I\alpha = 0.0029N \bullet m$$

[Rack & Pinion Bearings]

$$V_{gear} = 6.69 \times 10^{-9} m^3 \rightarrow m_{gear} = V_{gear} * 1410\ \frac{kg}{m^3} \approx 0.0094\ kg \rightarrow F_{gear} \approx 0.092N$$

$$with\ 2\ bearings,\ F_g = 0.046\ N\ per\ bearing$$

[Rack & Pinion Motor]

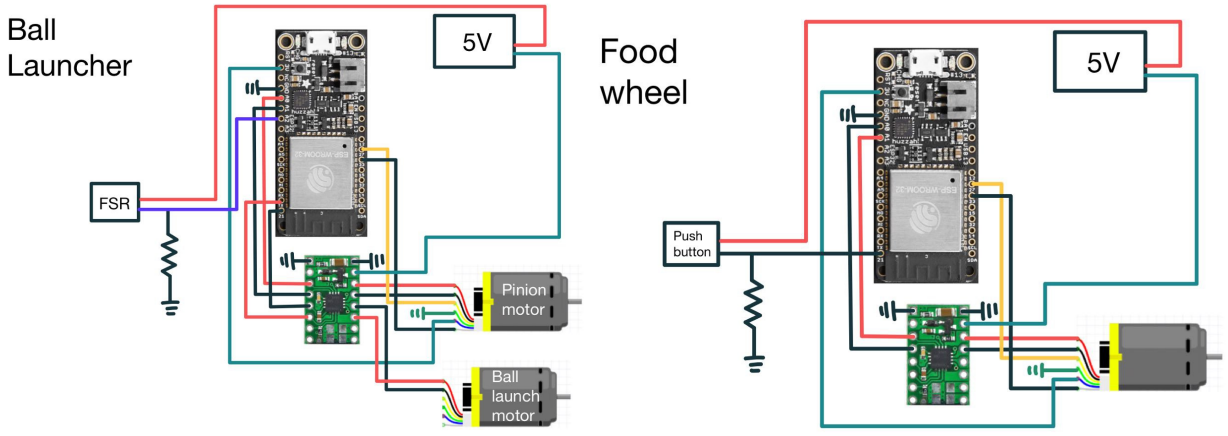
$$I_{pinion} = I_{hollow\ cylinder} = \frac{1}{2}m(r_o^2 + r_i^2) = \frac{1}{2}(.0.0094\ kg)(0.01125^2 - 0.0025^2) = 5.65 \times 10^{-7} kg \bullet m^2)$$

$$want\ to\ release\ ball\ in\ t = 3\ s$$

$$v = \frac{0.25m}{3s} = 0.83\frac{m}{s},\ \omega = \frac{v}{r} = \frac{0.083}{0.03m} \approx 2.78\ \frac{rad}{s},\ to\ reach\ full\ velocity\ in\ 1\ s,\ \alpha = \frac{\Delta\omega}{\Delta t} = 2.78\ \frac{rad}{s^2}$$

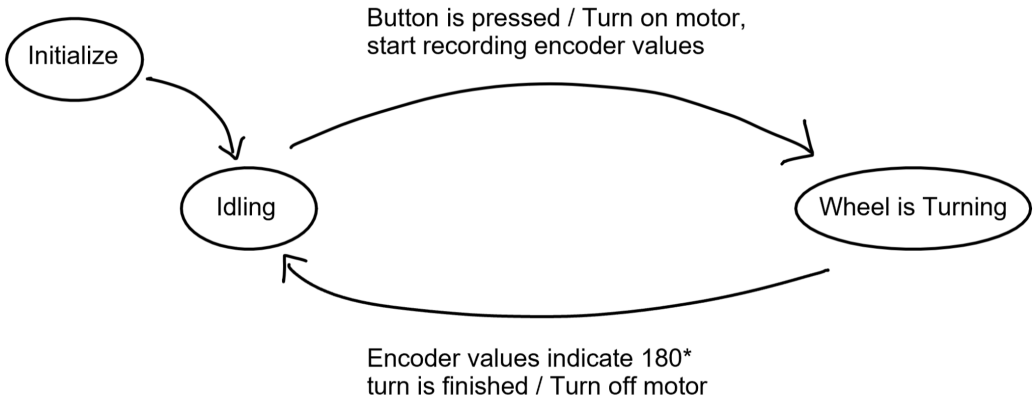
$$\tau = I\alpha = 1.57 \times 10^{-6} N \bullet m$$

Circuit diagrams:

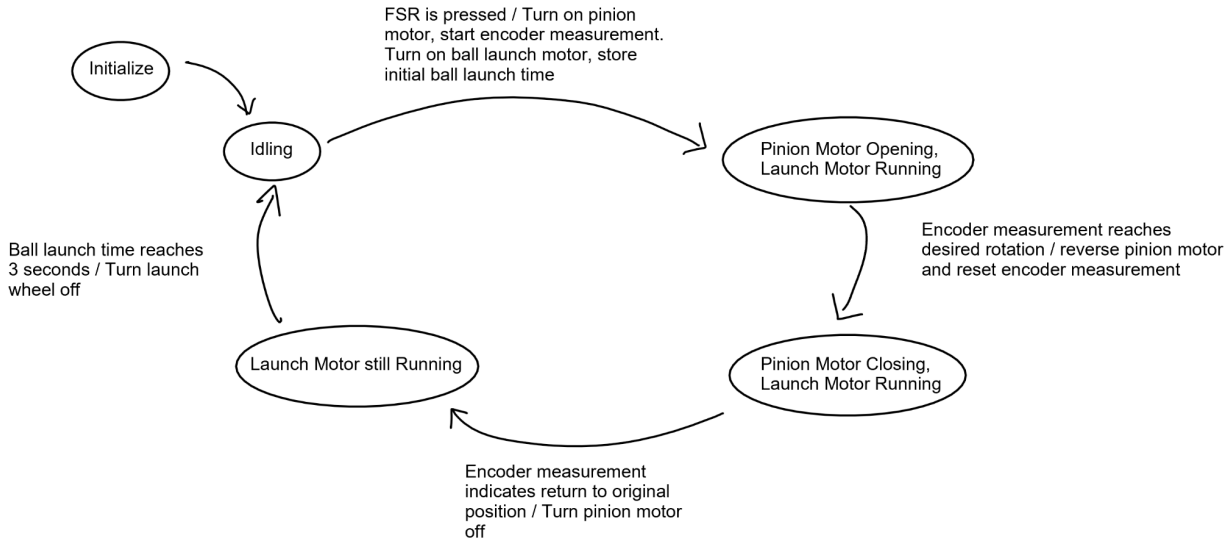


State Diagrams:

Food Wheel State Diagram:



Ball Launcher State Diagram:



Reflection:

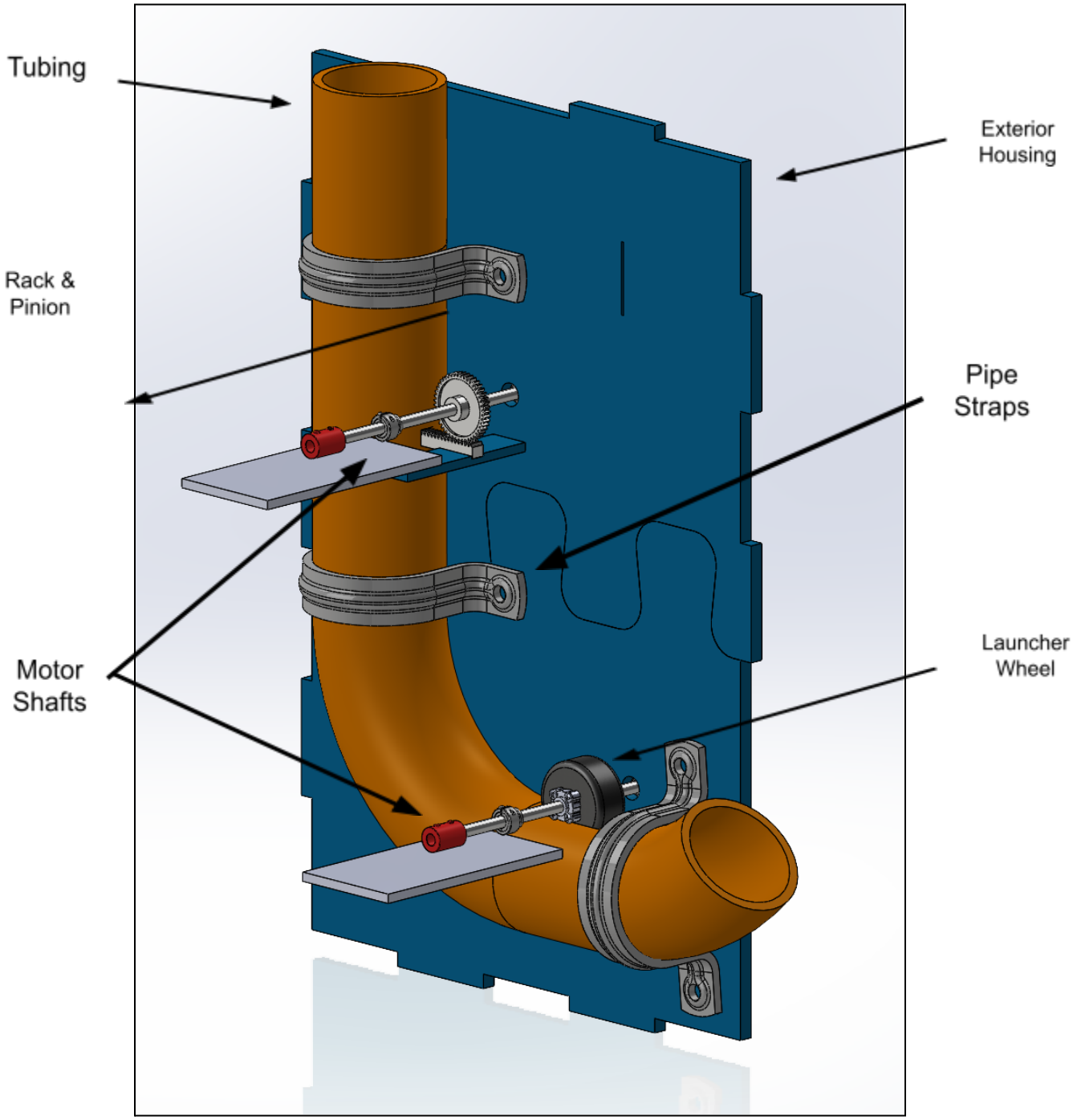
One thing that worked well for our group was having a set weekly meeting time that we could meet regularly for at least an hour and a half each week to keep us on track. At these meetings it was good to have an agenda planned beforehand and finish out with action items assigned to each person so everyone had something to work on for next time. Towards the end, it was really helpful to have a couple days where everyone was completely free and we were able to work on the project and finish assembling it all. In the future, it might be helpful to break up these major assembly days into a few four-hour chunks to maximize efficiency and reduce fatigue. Additionally, one challenge we faced was misalignment of our transmission and housing, which caused excess vibrations. With flexible shaft couplers being as expensive as they are, we had to figure out an alternative solution. With the help of the shop staff, we were able to create our own version of a flexible shaft coupler utilizing a piece of flexible plastic tubing and a 3D printed shaft converter.

Appendix A. Bill of Materials

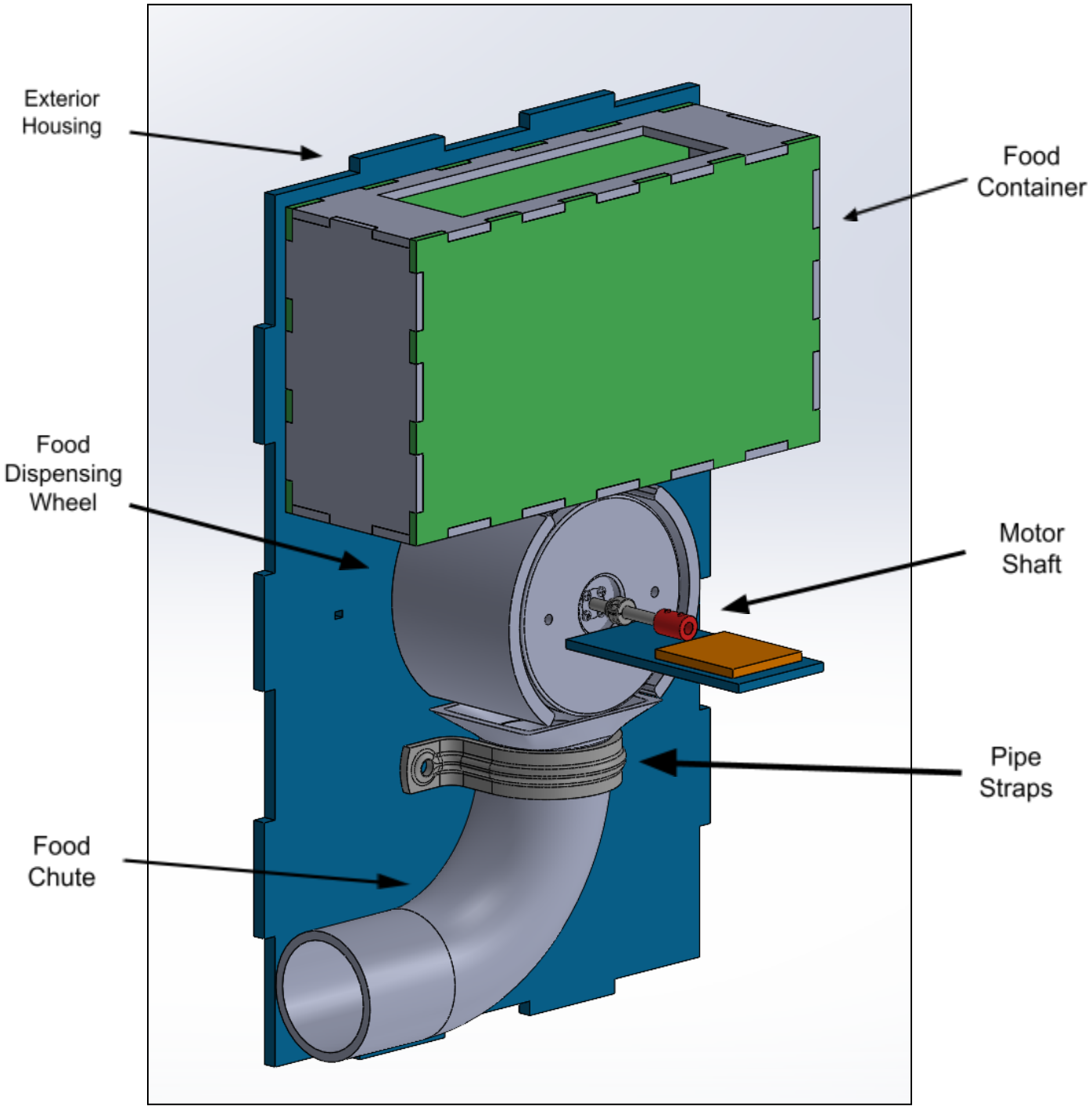
Name with Link	Product Number	Vendor	Quantity	Unit Cost	Total Cost	Ordered	Received
Need To Purchase							
8mm Shaft Collars (Pack of 10)		Amazon	1	$8.89	$8.89	☑	☑
18-8 Stainless Steel Round Shim, 0.5mm Thick, 8mm ID (pack of 25)	98089A336	McMaster	1	$7.96	$7.96	☑	☑
Beans		berkeley bowl	1	$5.00	$5.00	☑	☑
Belleville Disc Springs for Ball Bearing Trade No. 625, 634 and EI5, 8.2 mm ID (10 pack)	94065K34	McMaster	1	$4.00	$4.00	☑	☑
Shaft coupler 3mm to 8mm		Amazon	3	$11.49	$34.47	☑	☑
Tubing	5399K14	McMaster	3	$7.22	$21.66	☑	☑
Pipe hanger strap? / brackets		Home Depot	1	$6.46	$6.46	☑	☑
Custom Shafts		Misumi	3	$15.00	$45.00	☑	☑
Rack	2662N57	McMaster	1	$5.06	$5.06	☑	☑
Pinion	2662N342	McMaster	1	$8.17	$8.17	☑	☑
FSR		Adafruit	1	$5.95	$5.95	☑	☑
18x30" plywood (1/4 in thick)		Jacobs Store	9	$6.25	$56.25	☑	☑
8mm (0.700") Set Screw Hub	545636	Servocity	1	$5.99	$5.99	☑	☑
6-32 Pan Head Screws for Hub (pack of 25)	2817-0632-0500	Servocity	1	$2.19	$2.19	☑	☑
Motor for Ball Launcher	2363	Pololu	1	$19.95	$19.95	☑	☑
Bearings (10 pack)		Amazon	1	$10.00	$10.00	☑	☑
Compliant wheel		AndyMark	2	$8.61	$17.22	☑	☑
Wood Screws		Ace Hardware	80	$8.00	$8.00	☑	☑
L Brackets		Home Depot	1	$15.00	$15.00	☑	☑
Air Duct		Home Depot	2	$19.00	$38.00	☑	☑
Already Own							
Bread boards			3	$0.00	$0.00		☑
Wires			30	$0.00	$0.00		☑
Motors (from ME Microkit)			2	$0.00	$0.00		☑
Motor Drivers (from ME Microkit)			2	$0.00	$0.00		☑
Power Supply			1	$0.00	$0.00		☑
Button			1	$0.00	$0.00		☑
Food Storage Container			1	$0.00	$0.00		☑
Wiffle Ball			1	$0.00	$0.00		☑
Food Bowl			1	$0.00	$0.00		☑
				TOTAL COST:	$325.22		
				Taxed Total	$357.74		

Appendix B. CAD

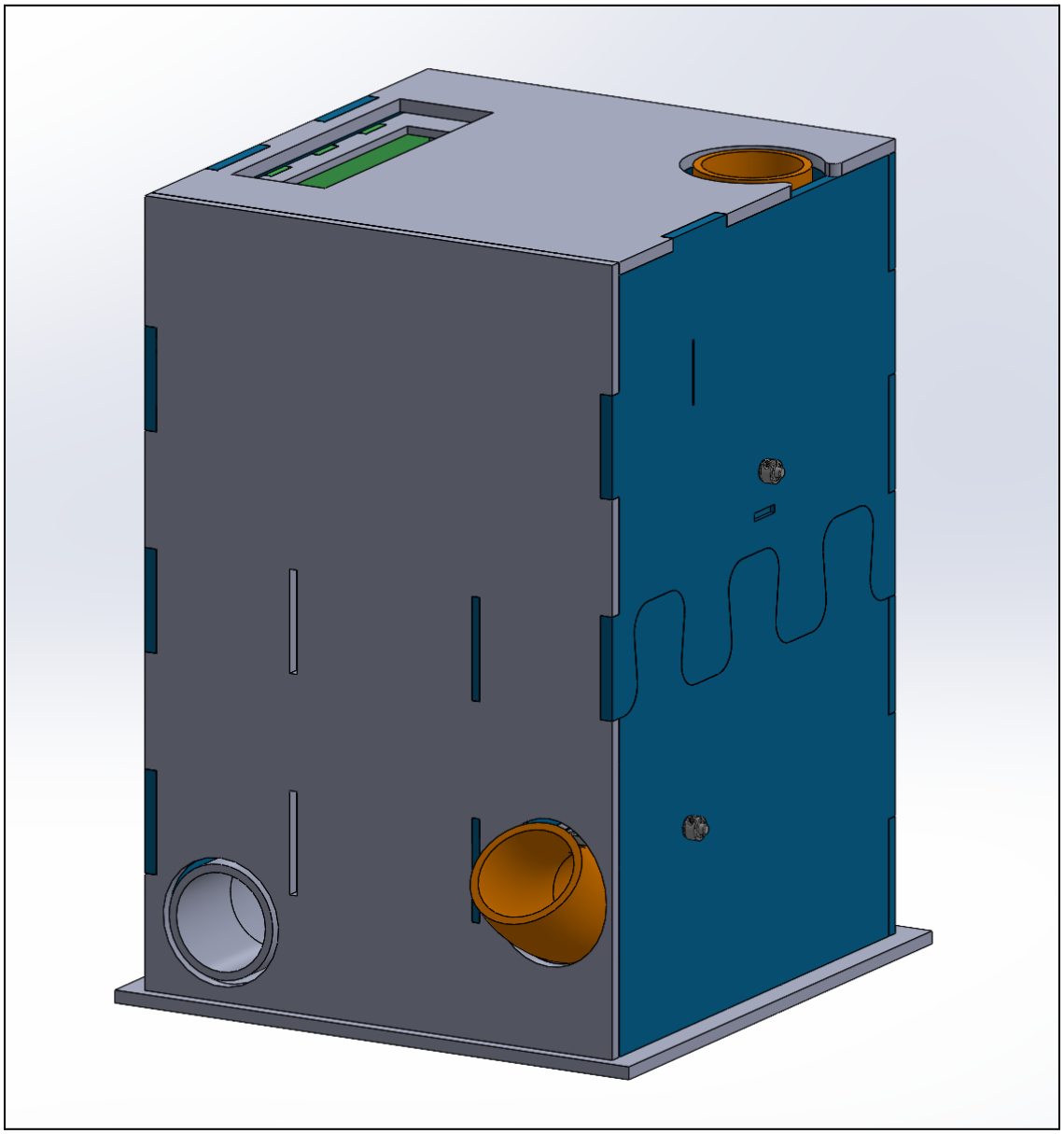
Ball Launcher Subsystem:



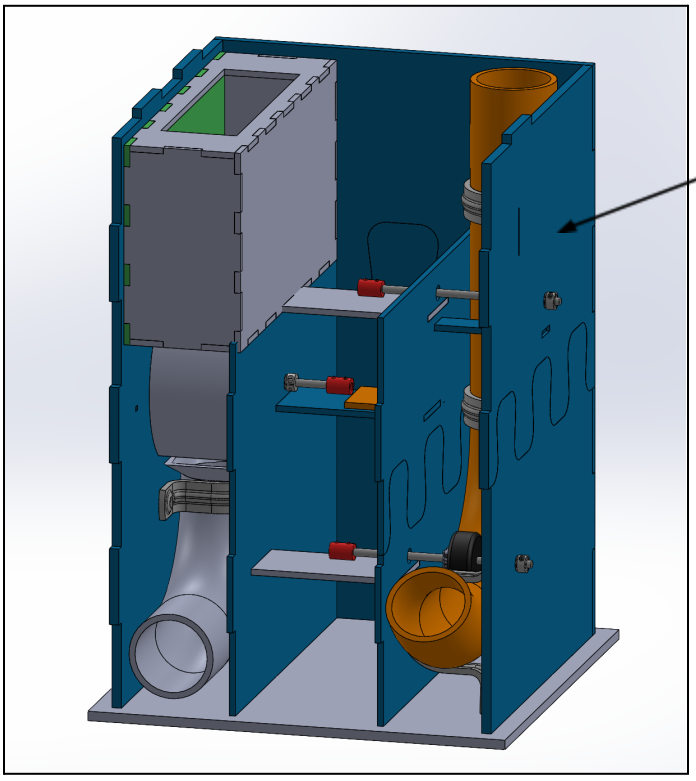
Feeder Subsystem:

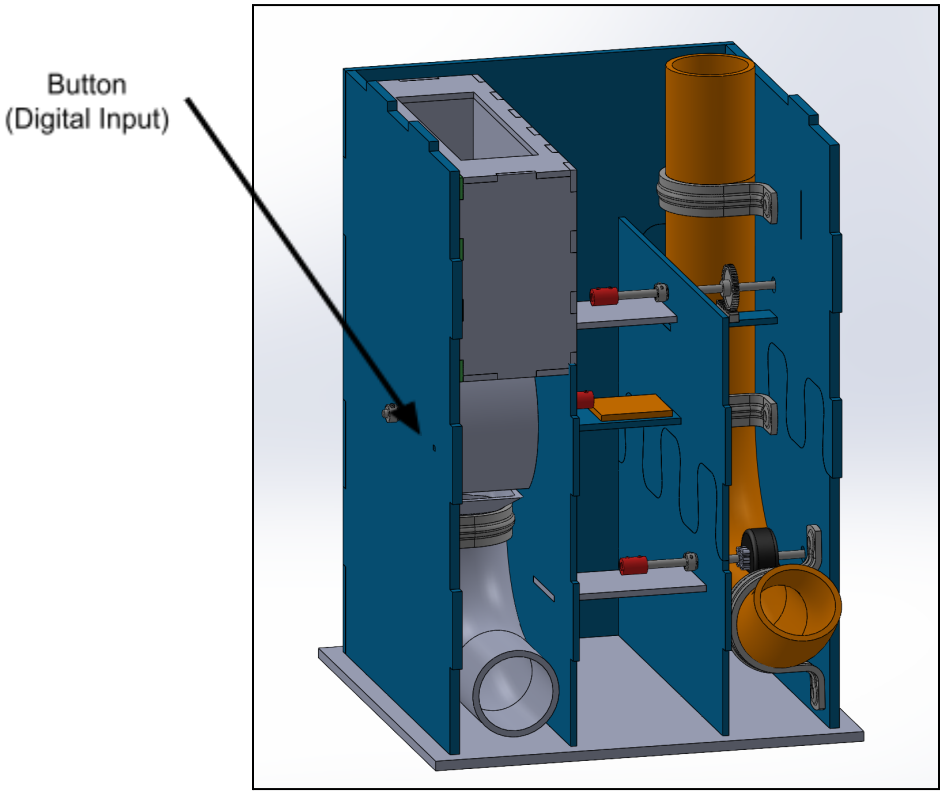


Full Assembly (w/ Full Housing):



Full Assembly (w/o Front & Top Housing):





Appendix C. Full Code

Food Wheel Code:

```
#include <Arduino.h>
#include <ESP32Encoder.h>

//Define constants ------------------------------------
#define BTN 21  // declare the button ED pin number
#define BIN_1 25
#define BIN_2 26

ESP32Encoder encoder;

//Setup variables -------------------------------------
int state = 1;
volatile int count = 0; // encoder count
volatile int basecount = 0;
volatile bool buttonIsPressed = false;
volatile bool debounceT = false;
hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

// setting PWM properties - motor -------------------------
const int freq = 5000;

const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int resolution = 8;
int MAX_PWM_VOLTAGE = 200;
int motor_PWM;

// encoder properties ------------------------------
int v = 0;
```

```
//Initialization -------------------------------------

void IRAM_ATTR onTime() {
  portENTER_CRITICAL_ISR(&timerMux);
  count = encoder.getCount( );
  encoder.clearCount ( );
  debounceT = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux);
  timerStop(timer);
}

void IRAM_ATTR isr() {  // the function to be called when interrupt is triggered
  buttonIsPressed = true;
  timerStart(timer);
}

void setup() {

  // put your setup code here, to run once:
  pinMode(BTN, INPUT);  // configures the specified pin to behave either as an input or an output
  attachInterrupt(BTN, isr, RISING);

  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

  //for motor configure LED PWM functionalitites
  ledcSetup(ledChannel_2, freq, resolution);
  ledcAttachPin(BIN_2, ledChannel_2);
  ledcSetup(ledChannel_1, freq, resolution);
  ledcAttachPin(BIN_1, ledChannel_1);
  //initialize timer
  timer = timerBegin(0, 80, true); // divides the frequency by the prescaler: 80,000,000 / 80 = 1,000,000 tics / sec
  timerAttachInterrupt(timer, &onTime, true);    // sets which function do you want to call when the interrupt is triggered
  timerAlarmWrite(timer, 500000, true);         // sets how many tics will you count to trigger the interrupt
  timerAlarmEnable(timer); // Enables timer
  timerStop(timer);

  Serial.begin(115200);

}

//Main loop ----------------------------------
void loop() {
  // put your main code here, to run repeatedly:
  switch(state) {
    case 1: //idling
      if (CheckForButtonPress()) { // button is pressed, move to state 2
        basecount = encoder.getCount(); // current encoder value
        MotorRun(); // run food wheel motor
        state = 2; // move to state 2
        //Serial.println("state 1 --> 2");
      }
      break;

    case 2: //turning food wheel
      //Serial.println(count);
      count = encoder.getCount(); // get encoder count to compare to reference
      if (CheckEncoder()){ // if encoder moved 180 degrees, change to state 1
        MotorOff(); // turn off motor
        state = 1; // move to state 1
        //Serial.println("state 2 --> 1");
      }
      break;
  }
}
```

```
// Event Checkers---------------------------------------

bool CheckForButtonPress(){ // check if feeder button is pressed
  if(buttonIsPressed && debounceT){
    //Serial.println("Pressed");
    debounceTimerReset();
    buttonIsPressed = false; // reset flag
    return true;
  }
  else {
    return false;
  }
}

bool CheckEncoder(){ // check if 180 degrees turned
  if (abs(count - basecount) >= 200){ // check if 180 degrees turned
    return true;
  }
  else{
    return false;
  }
}
```

```
// Services---------------------------------------
void MotorRun () { // runs motor to turn food wheel
  buttonIsPressed = false; // reset flag
  ledcWrite(ledChannel_2, LOW);
  ledcWrite(ledChannel_1, MAX_PWM_VOLTAGE);
}

void debounceTimerReset(){ // debouncing
  debounceT = false;
  timerStop(timer);
}

void MotorOff(){ // turns food wheel motor off
  ledcWrite(ledChannel_2, 0); //turn motor off
  ledcWrite(ledChannel_1, 0); //turn motor off
}
```

Ball Launch Code:

```
#include <Arduino.h>
#include <ESP32Encoder.h>

//Define constants ----------------------------------
//#define BTN 23  // declare the button ED pin number
#define BIN_1 26
#define BIN_2 25
#define AIN_1 17 // ball launch motor input
#define AIN_2 21
int FSR = 34;
int fsrReading;
unsigned long currentTime = 0;
unsigned long ogTime = 0;

ESP32Encoder encoder;

//Setup variables -----------------------------------
int state = 1;
volatile int count = 0; // encoder count
volatile int basecount = 0;
volatile bool FSRIsPressed = false;
volatile bool opening = false;
volatile bool debounceT = false;
hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

// setting PWM properties - motor ---------------------------
const int freq = 5000;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int ledChannel_3 = 4;
const int ledChannel_4 = 5;
const int resolution = 8;
int MAX_PWM_VOLTAGE = 100;
int MAX_PWM_VOLTAGE2 = 255;
int motor_PWM;
```

```
// encoder properties ------------------------------
int v = 0;

//Initialization -----------------------------------

void IRAM_ATTR onTime() {
  portENTER_CRITICAL_ISR(&timerMux);
  count = encoder.getCount( );
  encoder.clearCount ( );
  debounceT = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux);
  timerStop(timer);
}

void IRAM_ATTR isr() {  // the function to be called when interrupt is triggered
  FSRIsPressed = true;
  timerStart(timer);
}

void setup() {

  // put your setup code here, to run once:
  pinMode(FSR, INPUT);  // configures the specified pin to behave either as an input or an output
  attachInterrupt(FSR, isr, RISING);

  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

  //for motor configure LED PWM functionalitites
  ledcSetup(ledChannel_1, freq, resolution);
  ledcSetup(ledChannel_2, freq, resolution);
  ledcSetup(ledChannel_3, freq, resolution); // what input values do the channels need to be?
  ledcSetup(ledChannel_4, freq, resolution);
  ledcAttachPin(BIN_1, ledChannel_1);
  ledcAttachPin(BIN_2, ledChannel_2);
  ledcAttachPin(AIN_1, ledChannel_3);
  ledcAttachPin(AIN_2, ledChannel_4);

  //initialise timer
  timer = timerBegin(0, 80, true); // divides the frequency by the prescaler: 80,000,000 / 80 = 1,000,000 tics / sec
  timerAttachInterrupt(timer, &onTime, true);    // sets which function do you want to call when the interrupt is triggered
  timerAlarmWrite(timer, 250000, true);         // sets how many tics will you count to trigger the interrupt
  timerAlarmEnable(timer); // Enables timer
  timerStop(timer);
```

```
  Serial.begin(115200);

}

//Main loop ----------------------------------
void loop() {
  // put your main code here, to run repeatedly:
  switch (state) {
    case 1: //idling
      if (CheckForFSRPress()) { // FSR pressed, move to state 2
        basecount = encoder.getCount(); // get reference encoder count when FSR pressed
        RackOpen(); // rack/pinion motor on/opening
        WheelOn(); // start ball launch motor
        ogTime = millis(); // get current time for reference
        opening = true;
        //Serial.println("state 1 --> 2");
        state = 2; // move to state 2

      }
      break;

    case 2: //turning
      count = encoder.getCount(); // get encoder value to compare
      if (CheckEncoder()) { // check if rack/pinion has opened far enough
        basecount = encoder.getCount(); // get new reference encoder value for closing
        RackClose(); // set rack/pinion motor to closing
        opening = false;
        state = 3; // move to state 3
        //Serial.println("state 2 --> 3");
      }
      break;

    case 3: //turning
      count = encoder.getCount(); // get encoder value to compare
      if (CheckEncoderReturn()) { // check if rack/pinion closed far enough
        RackOff(); // stop rack/pinion motor
        state = 4; // move to state 4
        //Serial.println("state 3 --> 4");
      }
      break;

    case 4: //turning
      currentTime = millis(); // get current time to compare
      if (currentTime - ogTime > 3000) { // if 3 seconds has elapsed since FSR press
        WheelOff(); // turn off ball launch wheel motor
        state = 1; // move to state 1
        //Serial.println("state 4 --> 1");
        }
        break;
  }
}
```

```
// Event Checkers----------------------------------------

bool CheckForFSRPress() { // check if FSR pressed
  fsrReading = analogRead(FSR);
  //Serial.println(fsrReading);
  if ((fsrReading > 500)) {
    debounceTimerReset();
    FSRIsPressed = false;
    return true;
  }
  else {
    return false;
  }
}

bool CheckEncoder() { // check if encoder indicates rack/pinion opened far enough
  if (abs(count - basecount) >= 200) {
    return true;
  }
  else {
    return false;
  }
}
bool CheckEncoderReturn() { // check if encoder indicates rack/pinion closed far enough
  //count = encoder.getCount();
  //Serial.println(abs(count-basecount));
  if (abs(count - basecount) >= 150
     ) {
    return true;
  }
  else {
    return false;
  }
}
```

```
,
// Services----------------------------------------
void RackClose () { // close rack
  FSRIsPressed = false; // reset flag
  ledcWrite(ledChannel_1, LOW);
  ledcWrite(ledChannel_2, MAX_PWM_VOLTAGE);
}

void RackOpen () { // open rack
  FSRIsPressed = false; // reset flag
  ledcWrite(ledChannel_2, LOW);
  ledcWrite(ledChannel_1, MAX_PWM_VOLTAGE);
}

void RackOff() { // turn off rack/pinion motor
  ledcWrite(ledChannel_1, 0); //turn motor off
  ledcWrite(ledChannel_2, 0); //turn motor off
}

void debounceTimerReset() { // reset debounce flag
  debounceT = false;
  timerStop(timer);
}

void WheelOff() { // turn off ball launch wheel
  ledcWrite(ledChannel_3, 0); //turn motor off
  ledcWrite(ledChannel_4, 0); //turn motor off
}
void WheelOn() {
  ledcWrite(ledChannel_3, LOW); // turn on ball launch wheel
  ledcWrite(ledChannel_4, MAX_PWM_VOLTAGE2); //turn motor off
}
```