ME102B Project

Canberk Hurel - Aaron Wagner - Joshua Fong

Due 12/12/2021

1 Opportunity

Our opportunity for this project is that we want to make sure that pets are provided with sustenance in the case that the owners are away for an extended period of time and can not (or prefer not) to provide for the pets themselves.

2 High Level Strategy

Our pet feeder will be able to be operated both manually and automatically. The manual dispensing will be done through the buttons on the feeder. The user will be able to make a decision to enable the automatic feature by use of turning the potentiometer beyond the halfway mark. The user can also use the automatic food and water features independent of each other, for example one can use the automatic food state while maintaining manual water state, vice versa, having both manual food and water states, or both automatic food and water states. Finally, the user will be able to specify the amount of food or water dispensed at each cycle by adjusting the timer for which the door or solenoid valve is open or closed. In terms of the mechanical aspect, we are using a timing belt attached to two rotating pulleys with grooves to fit the teeth on the belt. We used springs on the timing belt in order to tension it, and two ball bearings in order to decrease radial load. Our initial design was very crude, using rubber bands and only one ball bearing with no tensioning of the belt/pulley system. We also initially wanted to drive the motor at a rate of 0.25 rev/sec, however this turned out to be too slow, as there was slippage between the timing belt and the pulley wheels. We increased the PWM duty to 200, which was sufficient for our needs.

3 Integrated Device

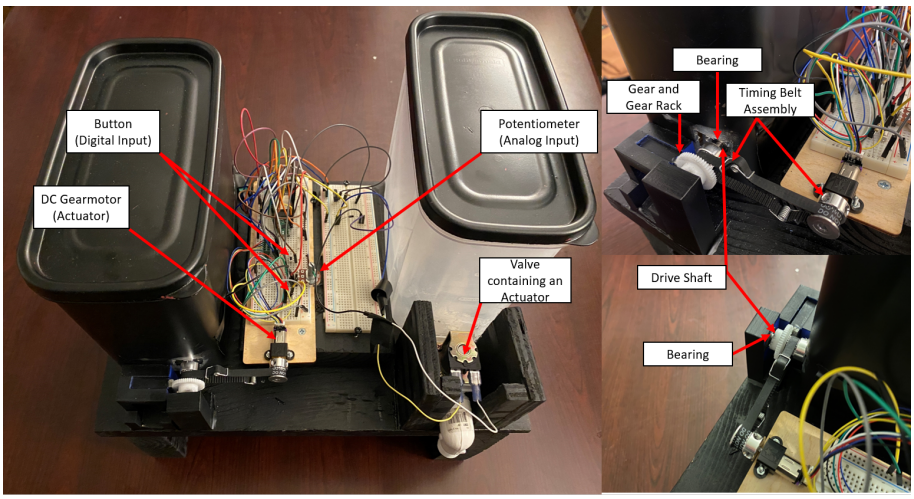

Figure 1: Completed device with a closeup of mechanical door system

4 Function-Critical Decisions

Our food dispensing mechanism is where we used a Micro Metal Gearmotor to operate a mechanical subsystem containing a timing belt system, a gear, a gear rack, two ball bearings, a shaft and a dispensing door made out of PLA. There were two important aspects we needed to figure out for the safe operation of this subsystem. The main goal of this subsystem is to lift the dispensing door upwards which weighs 17.5 g. In order to accomplish this goal, a 20 degree pressure angle gear rack is pasted onto the side of the door using epoxy and the movement created by the DC motor is transferred to the door using a 20 degree pressure angle gear. In this set-up our primary concern was related to the ability of our DC motor to provide the sufficient torque to lift the door. Investigating the interaction between the gear and the gear rack allowed us to determine the torque required to lift the gate. Considering the weight of the door, the gear would need to apply a tangent force, F_t, of 0.1717N. Through the gear force formula, the total required force can be found:

$$F_t = F_{total} \cdot cos(20) \tag{1}$$

From equation 1, the total required force is found to be 0.1827N. Considering that the gear has a radius of 1.25 cm, the torque required to operate this subsystem equals $0.1462N \cdot cm$. The DC motor used in the system is a Pololu 75:1 Micro Metal Gearmotor HP 6V with Extended Motor Shaft, with a nominal torque of $2.2563N \cdot cm$. Comparing this specification with the requirement indicates that the motor used is more than capable of providing the required torque.

The transfer of motion and force from the DC motor to the gear lifting the door was another important aspect of our subsystem. Initially, we planned to have a design with multiple of gears in order to not put any radial load on the motor shaft. However, due to the placement of other items and the availability of certain parts we decided to use a timing belt system to connect the gear with the DC motor. At this point, we made sure to analyze the radial force which would be put on the motor shaft. This radial force would be equal to the pre-tensioning force required for the timing belt. Remembering that the required input from the DC motor is $0.1462N \cdot cm$, the pre-tensioning force can be found through this relationship;

$$\tau = \frac{F_{pre}}{2} \tag{2}$$

From this formula F_{pre} is found to be 0.2924N, which is a minuscule amount of force to apply to the DC motor. After seeing this situation, we decided designing the mechanical subsystem with a timing belt would not cause any decline in the DC motor performance or durability. On the other side of the timing belt system, another timing pulley is placed on a shaft which also houses the gear used to lift the dispensing door. In order to stabilize this shaft both ends are placed in ball bearings which are also placed in support blocks created from PLA.

5 Diagrams

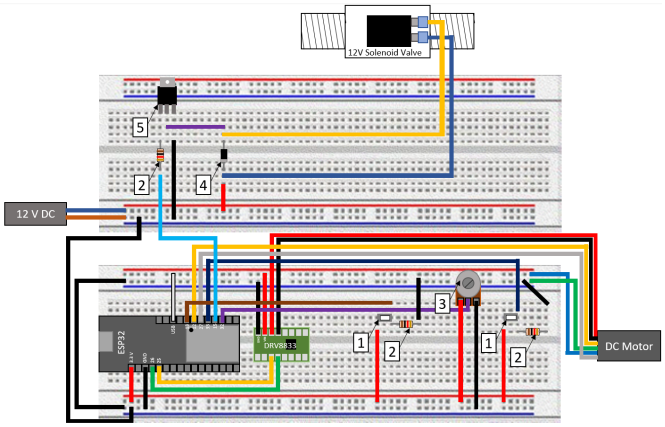

Figure 2: Circuit Diagram with (1) Buttons (2) 1 Kohm Resistors (3) Potentiometer (4) Diode (5) Transistor

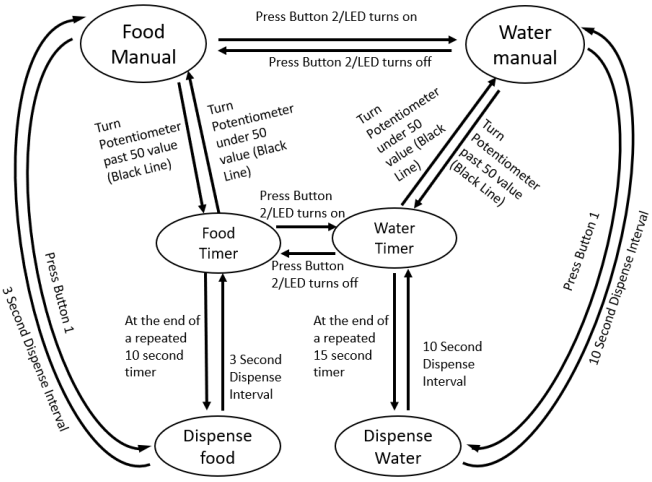

Figure 3: State Diagram of Machine

6 Reflection

Our project overall turned out adequate with the expectations that we had set in P1. We were pretty successful in debugging and overcoming obstacles within a short period of time. In hindsight, we felt that with more brainstorming, we could have come up with a more complex system that would have been doable within the allotted time frame.

7 Appendices

7.1 Appendix A: Bill of Materials

Bill of Materials

Item	Component name	Manufacturer/Source	Cost	Quantity
1	Plastic Water Solenoid Valve - 12V - 1/2" Nominal	https://www.adafruit.com/product/997#technical-details	6.95	1
2	12V DC 1000mA (1A) regulated switching power adapter	https://www.adafruit.com/product/798	8.95	1
3	ESP32 Microcontroller	Expressif Systems	provided	1
4	Motor+encoder with rotating shaft	unknown	provided	1
6	20 Degree Pressure Angle Plastic Gear, Round Bore, 48 Pitch, 48 Teeth	https://www.mcmaster.com/catalog/127/1231	3.89	1
7	20 Degree Pressure Angle Gear Rack, 48 Pitch	https://www.mcmaster.com/catalog/127/1231	3.89	1
8	PLA	Jacobs Makerspace	N/A	0.5 kg
9	Quick Set Epoxy	https://www.mcmaster.com/epoxies/quick-set-epoxy-structural-adhesives/	22.58	1 tube
10	Rubbermaid 16-Cup Dry Food Container Pack of 3	https://www.amazon.com/dp/B01HN89OBC?psc=1&ref=ppx_yo2_dt_b_product_	2.79	2
11	1/4" Ball Bearing Quad Pillow Block	https://www.amazon.com/Ball-Bearing-Quad-Pillow-Block/dp/B00KRJ7KUW	6.49	1
12	1/4" Shaft	https://www.amazon.com/Glarks-Stainless-Straight-Helicopter-6-35mmx356/dp/B	10.92	1
13	Gorilla Silicone Sealant	Ace Hardware	6.59	1
14	20 Teeth 4mm bore 6mm Width 20T Timing Belt Pulley Wheel Aluminum	https://www.amazon.com/WINSINN-Aluminum-Synchronous-Timing-Printer/	8.98	1
15	Double Metal Shielded Radial Ball Bearing	https://www.amazon.com/dp/B07P9H8R39?psc=1&ref=ppx_yo2_dt_b_product_d	8.5	1
16	20 Teeth 6.35mm Bore Timing Pulley Synchronous Wheel for 6mm Belt	https://www.amazon.com/dp/B07MGMBX3N?psc=1&ref=ppx_yo2_dt_b_produc	7.99	1
17	GT2 Timing Belt Torsion Spring for 3D Printer 6mm Width Belt	https://www.amazon.com/dp/B07L17GT5B?psc=1&ref=ppx_yo2_dt_b_product_d	3.99	1
18	3D Printer Timing Belt, 8 PCS 2GT-6 Closed Loop Rubber Belt	https://www.amazon.com/dp/B088M3V865?psc=1&ref=ppx_yo2_dt_b_product_d	10.99	1
19	50V 1A DO-41 Electronic Silicon Doorbell Diodes	https://www.amazon.com/dp/B071YWNBVM?psc=1&ref=ppx_yo2_dt_b_produc	4.59	1
20	3 Pieces TIP120 Power Darlington Transistors	https://www.amazon.com/dp/B00NAY1IBS?psc=1&ref=ppx_yo2_dt_b_product_d	7.99	1
21	Resistors (1k,10k) ohm	ME102B Kits	provided	2
22	Breadboard	ME102B Kits	provided	2
23	Push buttons	ME102B Kits	provided	2
24	Male-Male copper wires	ME102B Kits	provided	13
25	wood	Already in inventory	n/a	3 blocks
26	PVC piping (threaded)	Ace Hardware	2.49	2
27	screws	Ace Hardware	5.99	8
28	Potentiometer	ME102B Kits	provided	1

Figure 4: Bill of Materials

7.2 Appendix B: CAD

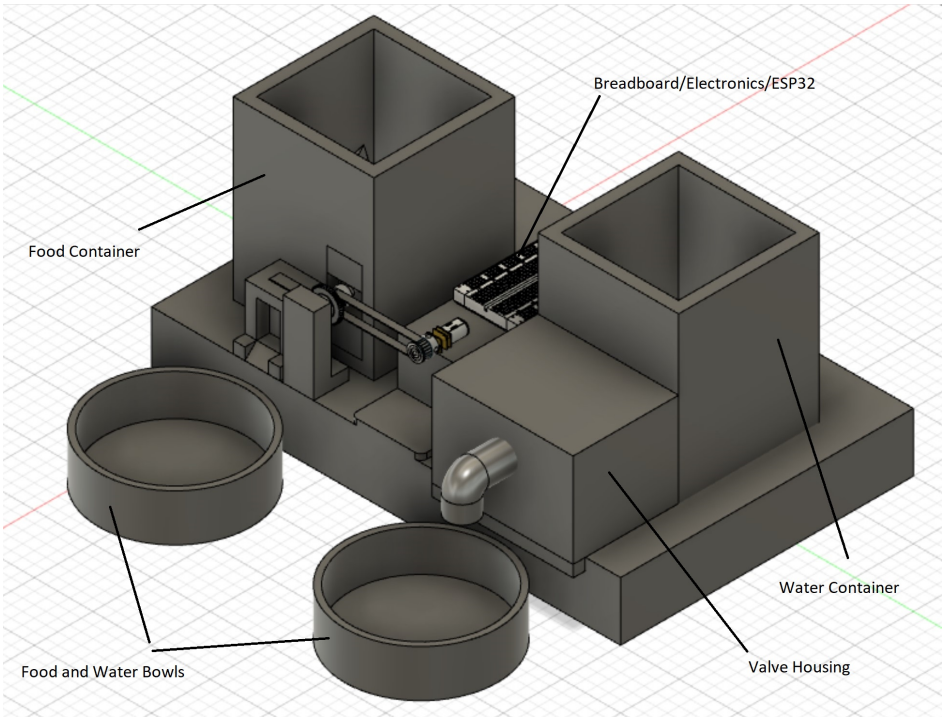

Figure 5: Isometric View with Valve Housing

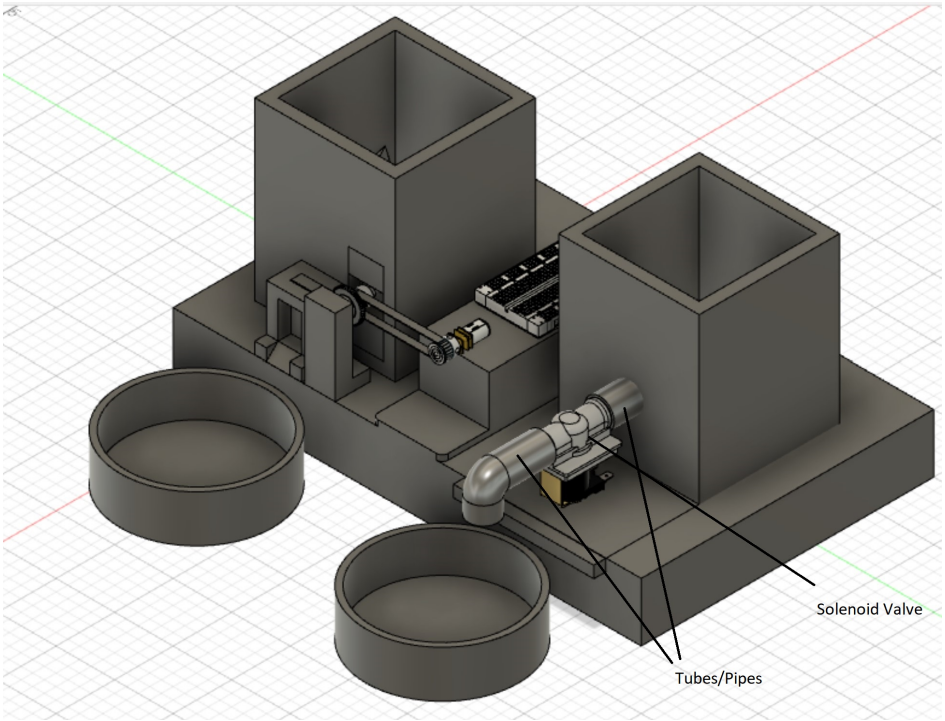

Figure 6: Isometric View without Valve Housing

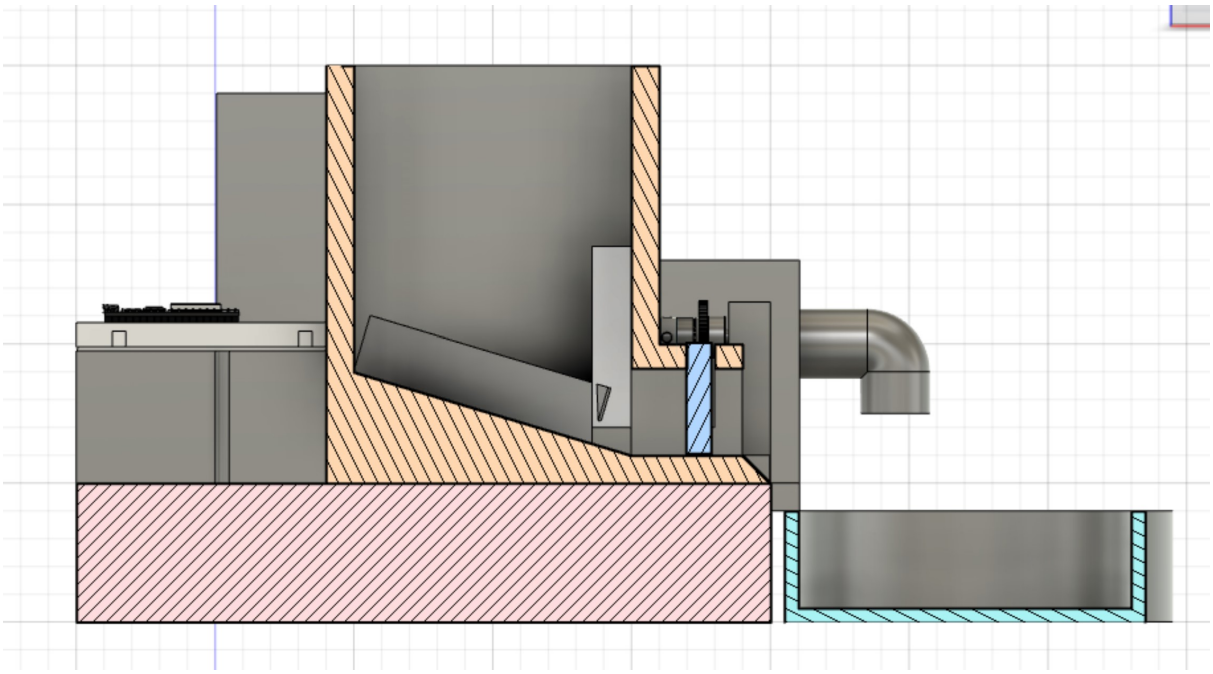

Figure 7: Cross-section View of Food Container

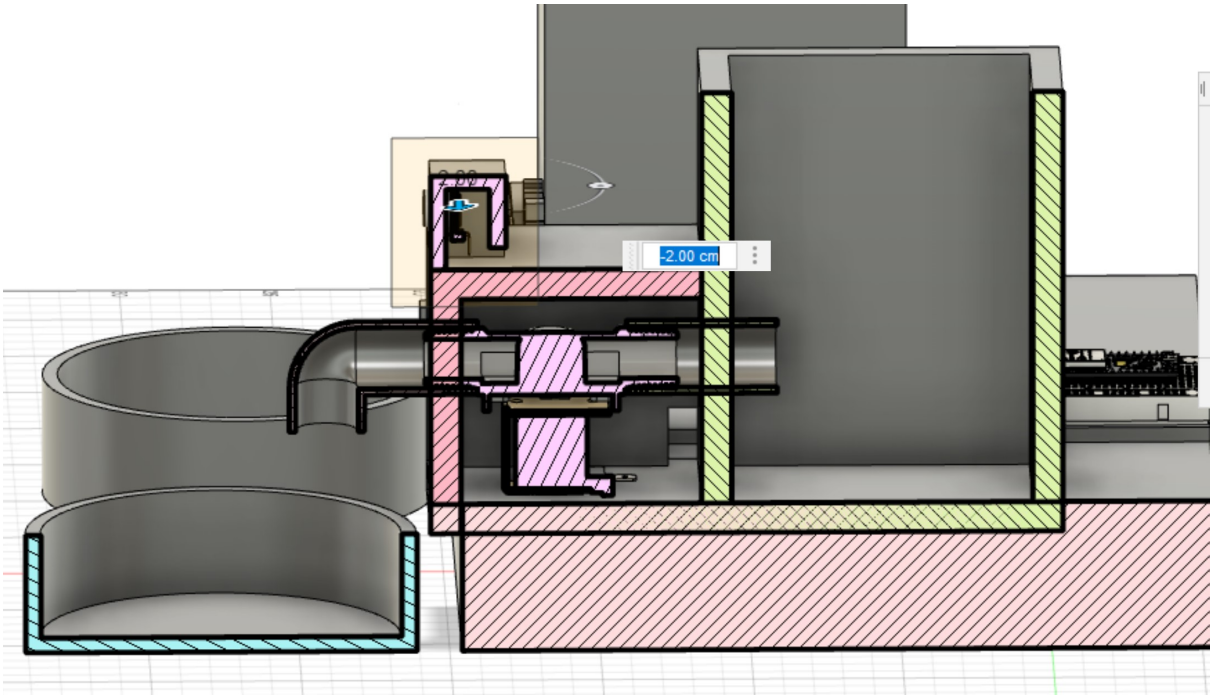

Figure 8: Cross-section View of Water Container

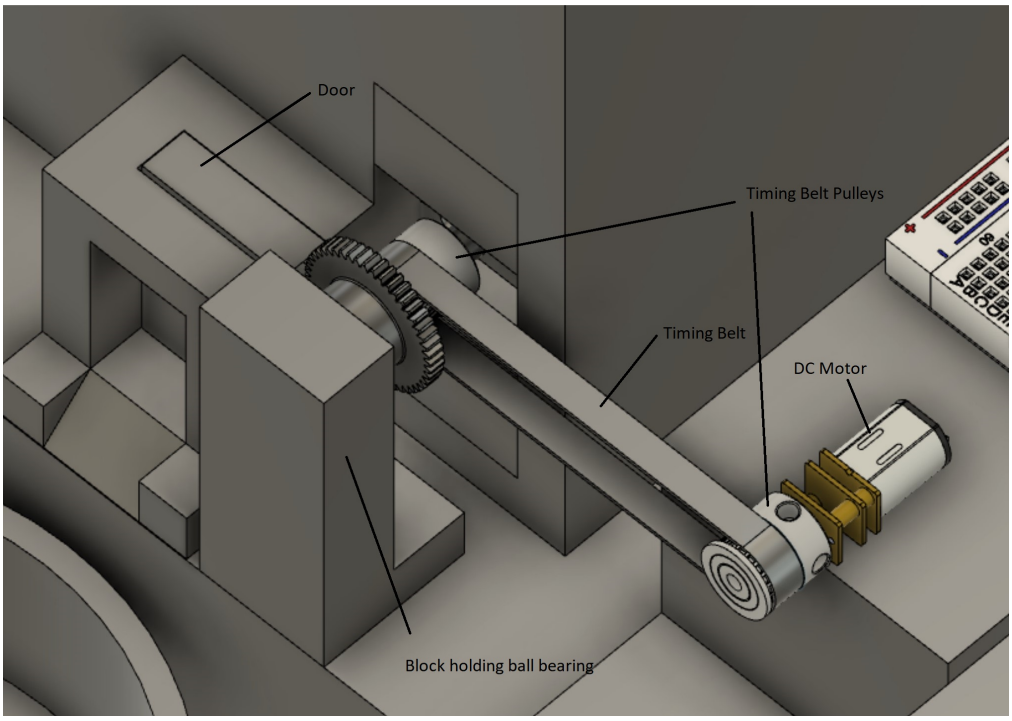

Figure 9: Close Up of food dispensing mechanism

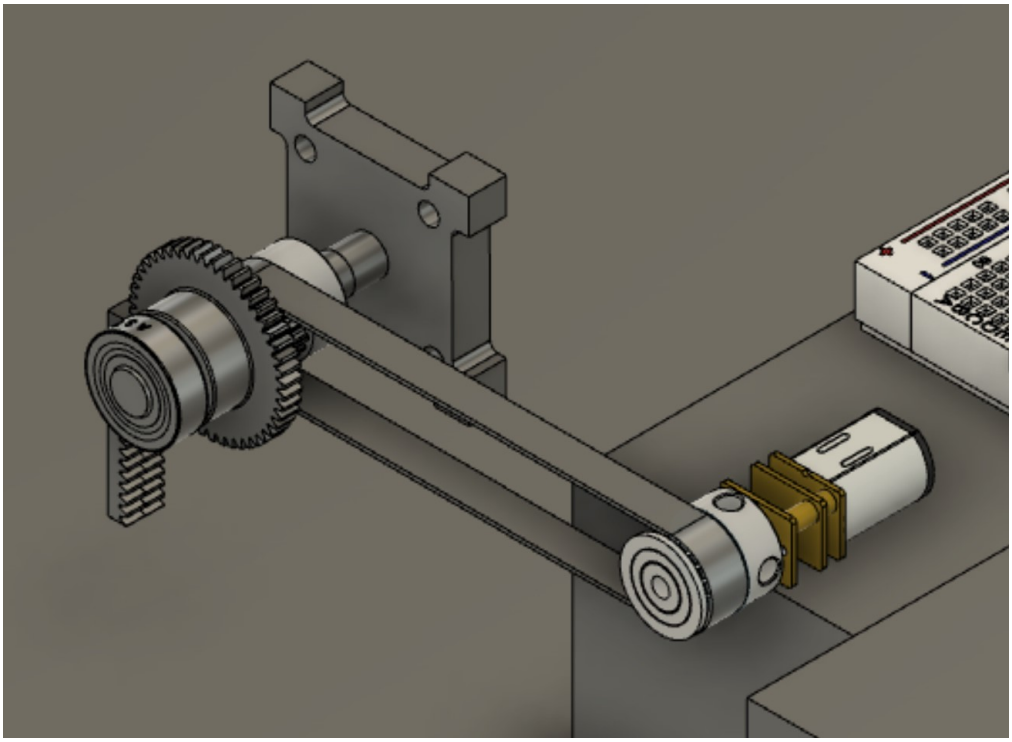

Figure 10: Close Up of food dispensing mechanism simplified

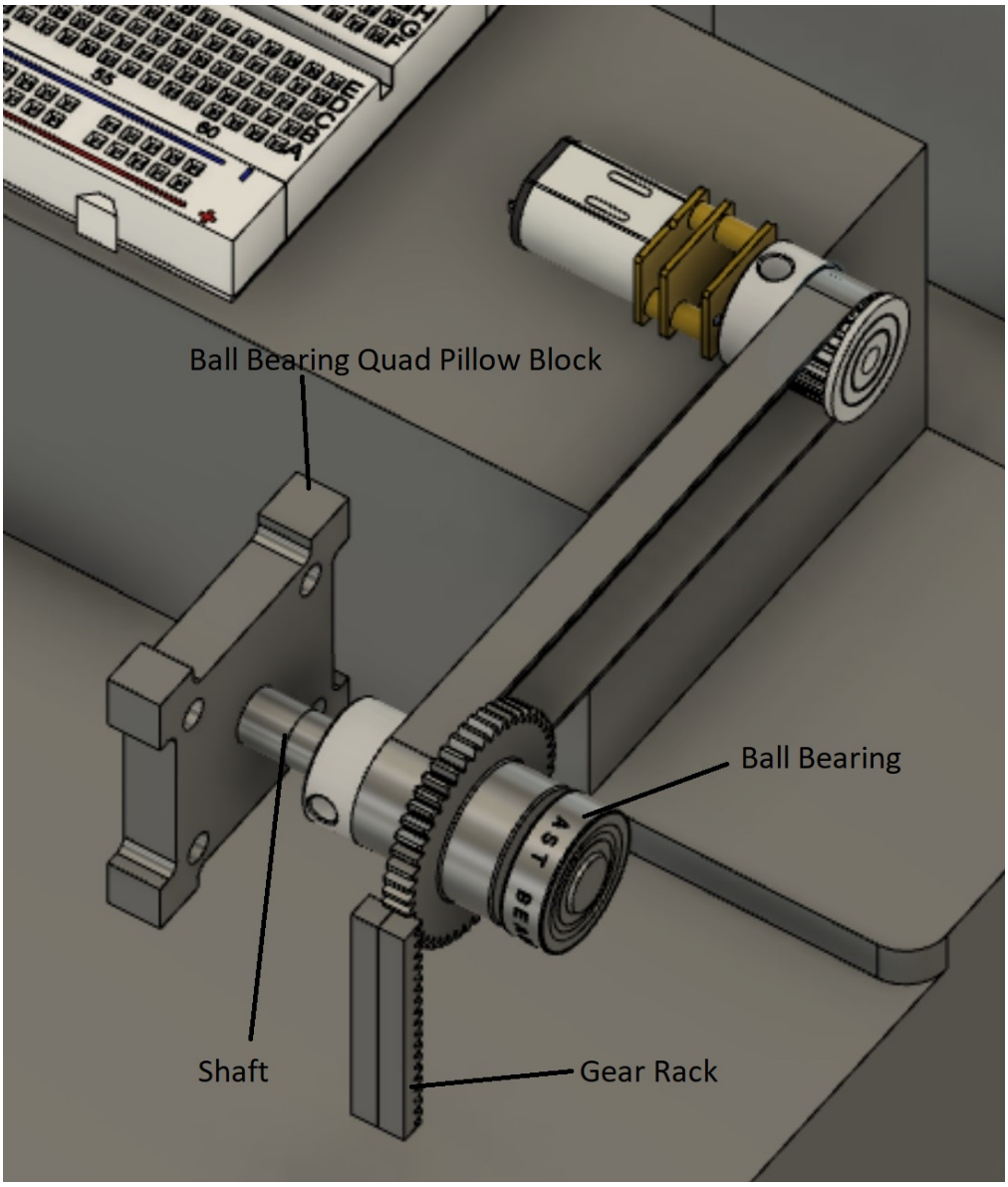

Figure 11: Close Up of food dispensing mechanism from opposite view

7.3 Appendix C: Code

```
#include <ESP32Encoder.h>
#define BIN_1 26
#define BIN_2 25
#define LED_PIN 13
#define BTN1 12
#define BTN2 15
#define POT 14
#define PUMP 32

ESP32Encoder encoder;

int D = 200;
int state = 0;
int potRead = 0;
int containerCounter = 0;
int foodTime = 1000000;
int waterTime = 0;

//Setup interrupt variables ----------------------------
volatile bool checkFoodTimer = false;    // check timer interrupt 1
volatile bool checkWaterTimer = false;     // check timer interrupt 2
int totalInterrupts = 0;   // counts the number of triggering of the alarm
hw_timer_t * timer0 = NULL;
hw_timer_t * timer1 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;
volatile bool button1IsPressed = false;
volatile bool button2IsPressed = false;

// setting PWM properties ---------------------------
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;

//Initialization -----------------------------------
void IRAM_ATTR onTime0() {
  portENTER_CRITICAL_ISR(&timerMux0);
  checkFoodTimer = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux0);
}
```

Figure 12: Code Snippet 1

```
void IRAM_ATTR onTime1() {
  portENTER_CRITICAL_ISR(&timerMux1);
  checkWaterTimer = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux1);
}

void IRAM_ATTR isr() {  // the function to be called when interrupt is triggered
  button1IsPressed = true;
}

void IRAM_ATTR isr2() {  // the function to be called when interrupt is triggered
  button2IsPressed = true;
}

void initFoodTimer() {
  timer0 = timerBegin(0, 80, true);  // timer 0, MWDT clock period = 12.5 ns * TIMGn_Tx_WDT_CLK_PRESCALE -> 12.5 ns * 80 -> 1000 ns = 1 us, countUp
  timerAttachInterrupt(timer0, &onTime0, true); // edge (not level) triggered
  timerAlarmWrite(timer0, 10000000, true); // 2000000 * 1 us = 2 s, autoreload true
  timerAlarmEnable(timer0); // enable
  Serial.println("Food timer initiated");
}

void initWaterTimer() {
  timer1 = timerBegin(1, 80, true);  // timer 1, MWDT clock period = 12.5 ns * TIMGn_Tx_WDT_CLK_PRESCALE -> 12.5 ns * 80 -> 1000 ns = 1 us, countUp
  timerAttachInterrupt(timer1, &onTime1, true); // edge (not level) triggered
  timerAlarmWrite(timer1, 10000000, true); // 10000 * 1 us = 10 ms, autoreload true
  timerAlarmEnable(timer1); // enable
  Serial.println("Water timer initiated");
}

void setup() {
  // put your setup code here, to run once:
  pinMode(BTN1, INPUT);// configures the specified pin to behave either as an input or an output
  pinMode(BTN2, INPUT);
  pinMode(POT, INPUT);
  pinMode(LED_PIN, OUTPUT);
  pinMode(PUMP, OUTPUT);
  digitalWrite(LED_PIN, LOW); // sets the initial state of LED as turned-off
  attachInterrupt(BTN1, isr, RISING);
  attachInterrupt(BTN2, isr2, RISING);
```

Figure 13: Code Snippet 2

```
  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);

  digitalWrite(PUMP, LOW);
}
```

Figure 14: Code Snippet 3

```
102 void loop() {
103   // put your main code here, to run repeatedly:
104   potRead = map(analogRead(POT), 0, 4095, 0, 100);
105   Serial.print("State: ");
106   Serial.println(state);
107   Serial.print("Potentiometer Value: ");
108   Serial.println(potRead);
109 //  Serial.println(checkFoodTimer);
110 //  Serial.println(checkWaterTimer);
111 
112   delay(1000);
113   switch (state) {
114 
115     case 0 : // food container is operated manually
116       if (checkWaterTimer) {
117         waterTimerExpireResponse();
118       }
119       if (button1PressEvent()) {
120         state = 2;
121         digitalWrite(LED_PIN, HIGH);
122         if (timer1 != NULL) {
123           timerStop(timer1);
124           Serial.println("Water timer stopped");
125         }
126       }
127       if (button2PressEvent()) {
128         state = 0;
129         dispenseFood();
130       }
131       if ( potRead >= 50 ) {
132         state = 1;
133         if (timer0 == NULL) {
134           initFoodTimer();
135         } else {
136           timerRestart(timer0);
137           Serial.println("Water timer restarted");
138         }
139       }
140       break;
```

Figure 15: Code Snippet 4

```
142     case 1: //Food timer
143       if (checkWaterTimer) {
144         waterTimerExpireResponse();
145       }
146       if (checkFoodTimer) {
147         foodTimerExpireResponse();
148       }
149       if (potRead < 50) {
150         state = 0;
151         timerStop(timer0);
152         Serial.println("Food timer stopped");
153       }
154       if (button2PressEvent()) {
155         state = 1;
156         digitalWrite(LED_PIN, LOW);
157         timerRestart(timer0);
158         Serial.println("Food timer restarted");
159       }
160       if (button1PressEvent()) {
161         state = 3;
162         digitalWrite(LED_PIN, HIGH);
163         if (timer1 == NULL) {
164           initWaterTimer();
165         } else {
166           timerRestart(timer0);
167           Serial.println("Water timer restarted");
168         }
169       }
170       break;
```

Figure 16: Code Snippet 5

```
171
172     case 2: // Water container operated manually
173       if (checkFoodTimer) {
174         foodTimerExpireResponse();
175       }
176       if (button1PressEvent()) {
177         state = 0;
178         digitalWrite(LED_PIN, LOW);
179         timerStop(timer0);
180         Serial.println("Food timer stopped");
181       }
182       if (button2PressEvent()) {
183         state = 2;
184         dispenseWater();
185       }
186       if (potRead >= 50) {
187         state = 3;
188         digitalWrite(LED_PIN, HIGH);
189         if (timer1 == NULL) {
190           initWaterTimer();
191         } else {
192           timerRestart(timer1);
193           Serial.println("Water timer restarted");
194         }
195       }
196       break;
```

Figure 17: Code Snippet 6

```
198     case 3: // Water timer activated
199       if (checkFoodTimer) {
200         foodTimerExpireResponse();
201       }
202       if (checkWaterTimer) {
203         waterTimerExpireResponse();
204       }
205       if (potRead < 50) {
206         state = 2;
207         digitalWrite(LED_PIN, HIGH);
208         timerStop(timer1);
209         Serial.println("Water timer stopped");
210       }
211       if (button2PressEvent()) {
212         state = 3;
213         digitalWrite(LED_PIN, HIGH);
214         timerRestart(timer1);
215         Serial.println("Water timer restarted");
216       }
217       if (button1PressEvent()) {
218         state = 1;
219         digitalWrite(LED_PIN, LOW);
220         if (timer0 == NULL) {
221           initFoodTimer();
222         } else {
223           timerRestart(timer0);
224           Serial.println("Food timer restarted");
225         }
226       }
227       break;
228
229     default: // should not happen
230       Serial.println("SM_ERROR");
231       break;
232   }
233 }
```

Figure 18: Code Snippet 7

```
234
235 //Event Checkers
236
237
238 bool button1PressEvent() {
239   bool button1State;
240   if (button1IsPressed == true) {
241     button1IsPressed = false;
242     button1State = true;
243     Serial.println("button 1 pressed");
244   } else {
245     button1State = false;
246   }
247   return (button1State);
248 }
249
250 bool button2PressEvent() {
251   bool button2State;
252   if (button2IsPressed == true) {
253     button2IsPressed = false;
254     button2State = true;
255     Serial.println("button 2 pressed");
256   } else {
257     button2State = false;
258   }
259   return (button2State);
260 }
261
```

Figure 19: Code Snippet 8

```
262 //Event Service Responses
263 void foodTimerExpireResponse() {
264   dispenseFood();
265   portENTER_CRITICAL(&timerMux0);
266   checkFoodTimer = false;
267   portEXIT_CRITICAL(&timerMux0);
268   timerRestart(timer0);
269   Serial.println("Food timer restarted");
270 }
271
272 void waterTimerExpireResponse() {
273   dispenseWater();
274   portENTER_CRITICAL(&timerMux1);
275   checkWaterTimer = false;
276   portEXIT_CRITICAL(&timerMux1);
277   timerRestart(timer1);
278   Serial.println("Water timer restarted");
279 }
280
281 void dispenseFood() {
282   Serial.println("Dispensing food");
283   ledcWrite(ledChannel_2, LOW);
284   ledcWrite(ledChannel_1, D);
285   delay(275);
286
287   ledcWrite(ledChannel_2, LOW);
288   ledcWrite(ledChannel_1, LOW);
289   delay(3000);
290
291   ledcWrite(ledChannel_1, LOW);
292   ledcWrite(ledChannel_2, D);
293   delay(190);
294
295   ledcWrite(ledChannel_2, LOW);
296   ledcWrite(ledChannel_1, LOW);
297 }
298
299 void dispenseWater() {
300   digitalWrite(PUMP, LOW);
301   delay(60000);
302   digitalWrite(PUMP, HIGH);
303   Serial.println("Dispensing water");
304 }
```

Figure 20: Code Snippet 9