Catch N' Treat: CaN'T

Group 27: Eric Gonzalez Bermudez, Migdalia Sanchez, Andrew P. Torri
Department of Mechanical Engineering, University of California, Berkeley
Effective Date: December 19th, 2024

Table of Contents:

I. Project Opportunity

Our project aims to fill the void in the hearts of cats and their owners. When cat parents are away from the home, the typical house cat is left with no one to play with. Cats may instead claw at furniture or create a mess, out of the frustration of no one to play with. Additionally, the common house cat can become quite overweight due to lack of activity. Our project, Catch N' Treat (CaN'T), aims to create a play and reward system to alleviate a cat's boredom,while increasing their physical activity. Unlike most generic cat toys which are stationary, and often require human involvement, CaN'T works by appealing to the cats natural hunting nature and rewarding this behavior to encourage further use.

II. High Level Strategy

The core of our project was to develop a fun and engaging cat toy that rewards the cat when playing with it. We put a cat toy on a load cell attached to a DC motor. The load cell acts as an analog input sensor to the ESP32, and triggers the treat dispensing mechanism when the signal from the load cell meets a threshold, indicating that the cat has successfully caught the treat. Alternatively, the treat can also be dispensed by the push of a button.

We wanted to launch the treat at a distance of 0.5 meters. The treats on average weighed approximately 0.25g / . We decided that a solenoid would be able to generate enough force to accomplish this task. As a part of the dispensing mechanism we also incorporated positional control on a DC motor to rotate a slotted disk that holds the treats. When the dispensing cycle is activated, a treat is dropped into the path of the solenoid by rotating the slotted disk by 45 degrees.

III. Integrated Physical Design

Our system is composed of a load cell, solenoid, DC motors, bearings, and gears. These were essential components to capture the transmission of our design. All electronic connections including our microcontrollers such as our ESP32 are protected by the wooden enclosure.

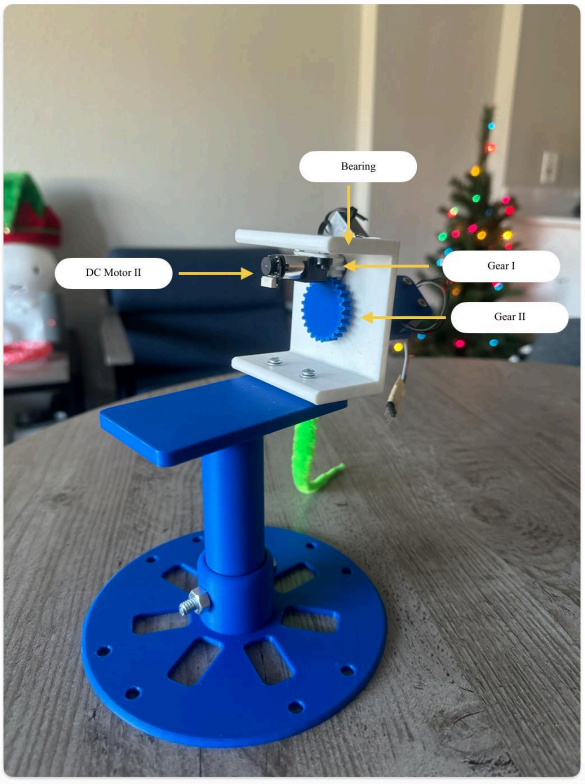
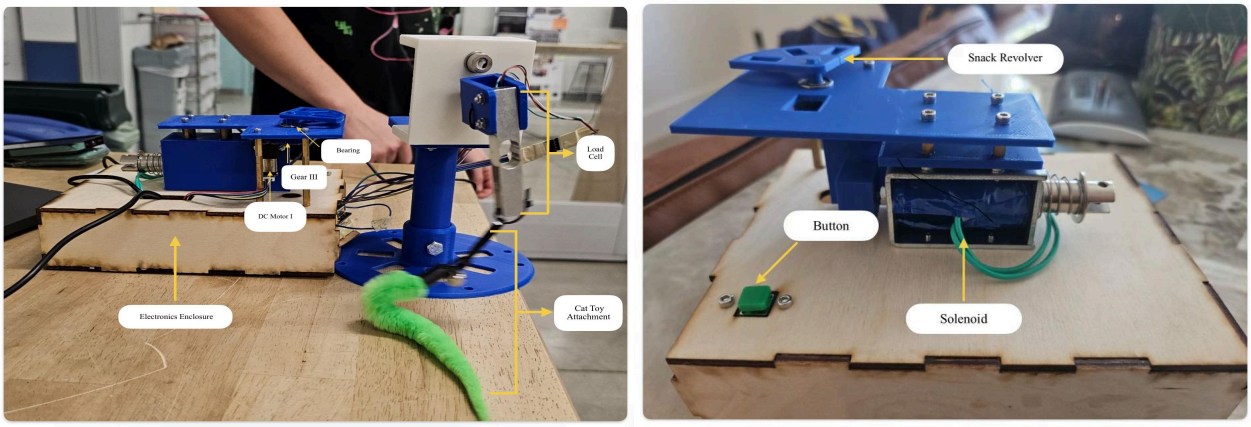


Figure 1: Load Cell and Rotation Design

The first section of our transmission is transferring the rotational motion of the DC motor to the load cell and cat toy attachment. As this rotates, once captured by the cat, it meets a force threshold which indicates the success of the capturing of the toy. In order to reward the cat for capturing the toy, as the threshold is met, it triggers the following treat dispensing system. The dispensing system is a connection of a treat revolver and DC motor. A bearing was used to optimize rotation and reduce friction between the motor and snack rotation components.

Final section of the integrated design is the solenoid actuation to launch the treat. Once the snack rotation reaches the opening design for its fall, the solenoid is initiated for snack launching. The crucial aspect of this design is having enough voltage to supply the solenoid and motors. If the voltage is not sufficient, the solenoid will not function properly and the cat will not be rewarded for capturing the toy.



[Figure 2: Integrated and Close Up of Integrated Physical Design]

IV. Functional Critical Designs and Calculations

Rotating Cat String Gearbox

Our first critical design consideration was the load on the motor when the cat pulls on it. We estimated roughly 10N-20N output from the cat that would be absorbed into the shaft. The following calculation reports the torque applied to the base of the motor and falls within the max allowable .

Torque = (10 N)* 0.0254m = (.254N/m).

Solenoid Launcher

Another critical component of the project was the ability to launch the treat a considerable distance. The distance we considered significant was about 6". Other information we had available to us was treat mass (~0.25g), height of launch pad (3"). Our goal is to achieve an approximate velocity of 5m/s starting with an initial velocity of 0 m/s within the range of 10-100ms. With these knowns and assumptions, we can tease out a force required to launch the treat by using basic kinematic relationships. Starting with Newton's 2nd law:

$$F = ma$$

Where,

$$a = (v_2 - v_1)/(t_2 - t_1).$$

Then the force needed for the snack to obtain the desired velocity is calculated by,

$$F = (0.25g)(5m/s - 0m/s)/(0.05s - 0s) = \mathbf{25\ mN}$$

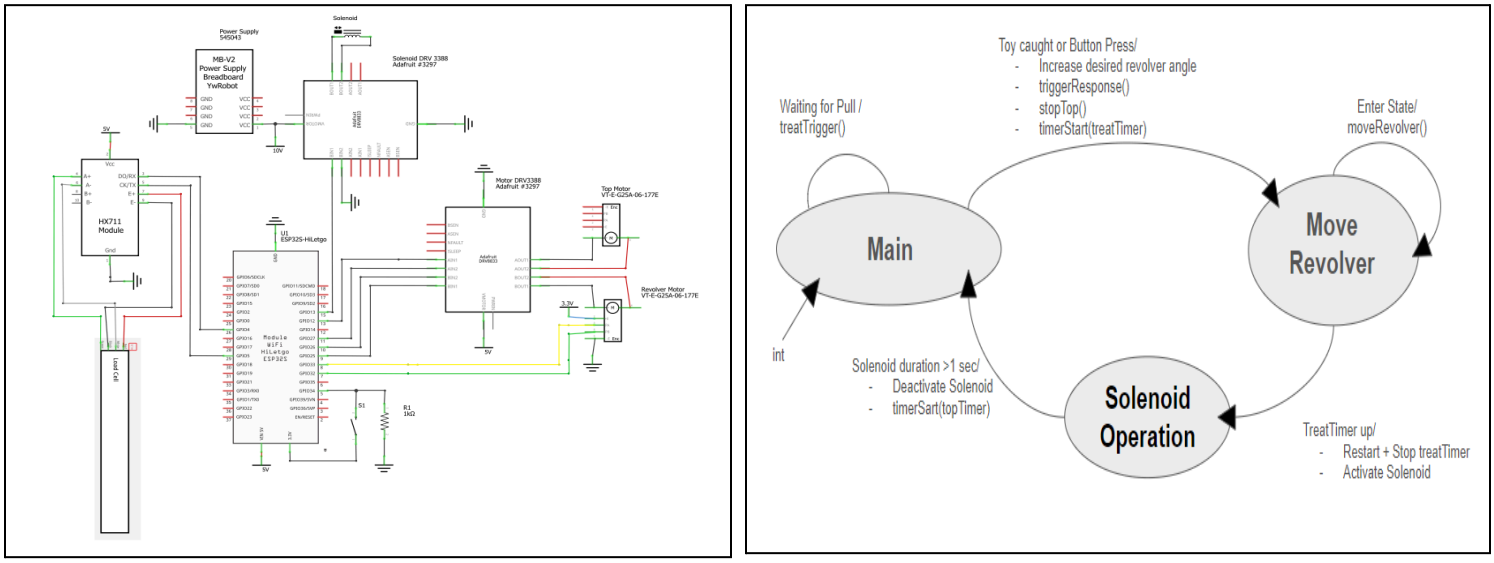
Revolver Motor Power

The final consideration dealt with the ability for the motor to reliably rotate the revolver. Since we decided to 3D print the revolver, we modeled the revolver as a solid PLA cylinder with dimensions of height (10mm) and radius (35mm). Then the max torque this motor could experience under a moderate angular acceleration of roughly 3 radians/s^2.

$$\tau = I\alpha = \frac{1}{2}(0.25g)(35x10^{-3}\,m)^2(3\,\frac{rad}{s^2}) = 4.6\,x10^{-4}\,\frac{gm^2}{s^2}$$

This value falls far below the motor's load torque of 1000 g*cm.. However we had an issue with the dead zone of the motor. When we tried using positional control, the input voltage to the motor was far too small. To get around this, we opted to use a 2:1 gear ratio to reduce the amount of radians per count of the motor encoder.

V. Circuit and State Machine Diagram

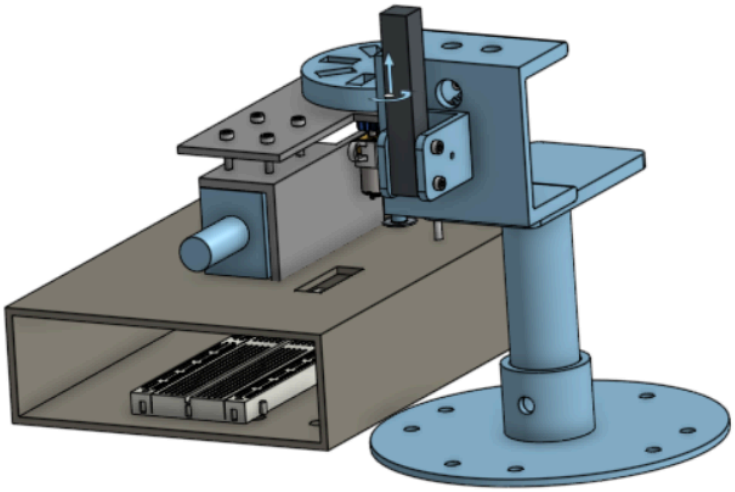


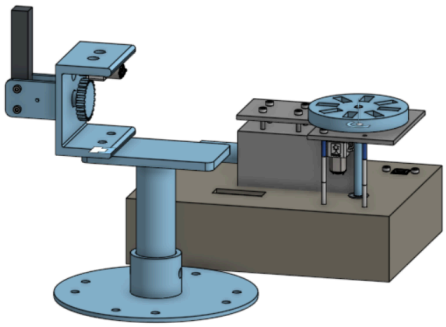
[Figure 3: CaN'T Circuit & State Machine Diagram (Appendix D)]

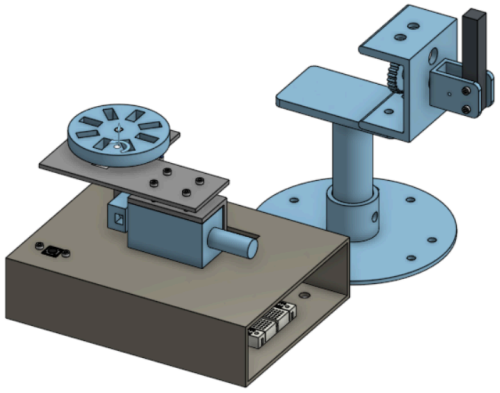
VI. Reflection

Establishing a design, creating a prototype design, and achieving a final design within one semester is a difficult challenge. Critical aspects for our project's success was maximizing each teammate's skills and resources such as access to 3D printing, coding experience, and team accountability to complete deliverables. Challenges were scheduling work sessions and creating a cohesive integrated system apart from our subsystems. One advice for future students is to help each other co-create a design together with subsystems rather than allocating one individual team member to take this on. This aspect can help everyone be on the same page and communicate with one another throughout the design and manufacturing process.

Appendix A : CAD Designs







Appendix B : Bill of Materials

category	Part/ Material	Unit Price	Quantity	Unit Cost	Status	Link
Mechanical	Solenoid, Tubular , Pull	9.72	1	9.72	acquired	Amazon
	Screw, M3 Screw Kit	8.99	1	8.99	acquired	Amazon
	Standoffs, M3 Hex Kit	11.99	1	11.99	acquired	Amazon
	Ball Bearing, Flanged	9.29	1	9.29	acquired	Amazon
	Button, Tactile	5.95	1	5.95	acquired	DigiKey
	Plywood, 1/8" x 18" x 30"	8.32	1	8.32	acquired	Jacobs
Electrical	ESP32	20.95	1	20.95	acquired	Digikey
	10V / 2A Power supply	17.99	1	17.99	acquired	Amazon
	BREADBOARD POWER 3.3V AND 5V	6.3	1	6.3	acquired	Digikey
	DC MOTOR WITH MAGNETIC ENCODER	12.05	2	24.1	acquired	Digikey
	HX 711 Load Cell	7.98	1	7.98	acquired	Amazon
	DRV8833 Dual Motor Driver Carrier	9.95	2	19.9	acquired	Pololu
	Breadboard	5.95	2	11.9	acquired	DigiKey
	Resistor, 10k ohms	0.12	1	0.12	acquired	Digikey
	Jumper Wires	2.1	1	2.1	acquired	DigiKey
3D Printed	Tower Stage	1.5	1	1.5	acquired	In house CAD
	Launcher housing	1	1	1	acquired	In house CAD
	Gear, 3in	1.5	1	1.5	acquired	in house CAD
	Gear, 1 in	0.5	1	0.5	acquired	in house CAD
Misc.	Cat toy	3.99	1	3.99	acquired	PetCo
Project total:	**174.09**					

Appendix C : CaN'T code

```
#include <Arduino.h>
#include <ESP32Encoder.h>
#include "HX711.h"    // library for force sensor
// HX711 Force sensor object
HX711 scale;
long reading = 0 ; // Reading from hx711
#define DT 4          // Data pin connected to HX711
#define SCK 5         // Clock pin connected to HX711

// Setup encoder object, H-bridge assign, and variables for revolver motor
ESP32Encoder revolverEncoder;
#define BIN_1 25      // B in 1 for Revolver motor
#define BIN_2 26      // B in 2 for Revolver motor
int desPOS = 0 ;      // desired position of revolver
int currPOS = 0 ;     // current position of revolver
int error = 0 ;
int inc45 = map(90, 0, 360, 0, 455);  // 455 counts per revolution
int D_revolve = 0 ;  // initial pwm voltage
int Kp =  15; // proportional gain for the snack revolver motor (divided
by 10 later
int Ki = 0  ; // integral gain for the snack revolver motor (divided by
100 later)
int I = 0 ; // I term for revolver PI controller
int P = 0 ; // P term for revolver PI controller
int e_sum = 0 ;
volatile bool motorIsOFF = false ;


// declare button pin number
#define BTN 34


// Top motor pins and variables
#define AIN_1 12
#define AIN_2 27
volatile bool motorDirection = false ;


// Variables for Solenoid and actuation
#define SOL 13  // Declare pin for solenoid
unsigned long currentTime = 0;
unsigned long lastCycleTime = 0;
unsigned long elapsedTime = 0 ;
const int cycleDuration = 1000;


const int MAX_SNACK_MOTOR_VOLTAGE = 150;


// pwm properties
const int freq = 21000;
```

```
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int resolution = 8;


// timers
hw_timer_t * treatTimer = NULL;
hw_timer_t * topTimer = NULL;
hw_timer_t * debounceTimer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;


volatile bool buttonIsPressed = false;
volatile bool debounceFlag = false;
volatile bool directionFlag = true;
volatile bool triggerFlag = false;
volatile bool treatFlag = true;
volatile bool solTrigger   = false ;
int angle = 0 ;
int state = 1 ;
volatile int count = 0; // encoder count


//#########################################################################
//################    Interrupts  #######################################
//#########################################################################


void IRAM_ATTR topTime() {
  portENTER_CRITICAL_ISR(&timerMux);

  if (motorDirection) {
    digitalWrite(AIN_1, LOW);  // BIN_1 off
    digitalWrite(AIN_2, HIGH); // BIN_2 drives motor in reverse
    motorDirection = !motorDirection;
  } else{
    digitalWrite(AIN_1, HIGH);  // BIN_1 off
    digitalWrite(AIN_2, LOW);   // BIN_2 drives motor in reverse
    motorDirection = !motorDirection;
  }
  portEXIT_CRITICAL_ISR(&timerMux);
}


// Timer interrupt to initiate solenoid mechanism and move into state 3
// 1. Timer is set long enough to ensure revolver rotates 45 degrees
void IRAM_ATTR treatTime() {
  portENTER_CRITICAL_ISR(&timerMux1);
  state = 3;
  timerRestart(treatTimer);
  timerStop(treatTimer);
  lastCycleTime = millis();
  digitalWrite(SOL,HIGH);
  portEXIT_CRITICAL_ISR(&timerMux1);
```

```
}

// Timer interrupt for button debounce
void IRAM_ATTR bounceTime() {
  portENTER_CRITICAL_ISR(&timerMux1);
  debounceFlag= true;    buttonIsPressed = false;
  timerStop(debounceTimer);
  portEXIT_CRITICAL_ISR(&timerMux1);

}

// the function to be called when interrupt is triggered
void IRAM_ATTR button_isr() {
  if(debounceFlag){
    buttonIsPressed = true;
    debounceFlag =false;
    Serial.println(buttonIsPressed);
    timerStart(debounceTimer);
  }
}

// ##########################################################################
// ################    SETUP    ###############################################
// ##########################################################################

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(BTN, INPUT);

  // pin for solenoid
  pinMode(SOL, OUTPUT);

  // pins for REVOLVER motor
  pinMode(BIN_1, OUTPUT);
  pinMode(BIN_2, OUTPUT);
  ledcAttach(BIN_1, freq, resolution);
  ledcAttach(BIN_2, freq, resolution);

  // pins for TOP motor
  pinMode(AIN_1, OUTPUT);
  pinMode(AIN_2, OUTPUT);

 // timer for trigger
  treatTimer = timerBegin(1000000);
  timerAttachInterrupt(treatTimer, &treatTime);
  timerAlarm(treatTimer, 1000000, true,0 );
  timerStop(treatTimer);

  // Timer for TOP motor
```

```
  topTimer = timerBegin(1000000);
  timerAttachInterrupt(topTimer, &topTime);
  timerAlarm(topTimer, 300000, true,0 );

  // Debounce timer
  debounceTimer = timerBegin(1000000);
  timerAttachInterrupt(debounceTimer, &bounceTime);
  timerAlarm(debounceTimer,50000,true,0);
  timerStop(debounceTimer);

  // button pin and interrupt
  pinMode(BTN, INPUT);
  attachInterrupt(BTN, button_isr, RISING);

  // Setup for force sensor
  scale.begin(DT, SCK);
  scale.tare();

  // Encoders
  // Enable the weak pull up resistors
  ESP32Encoder::useInternalWeakPullResistors = puType::up;
  // Attache pins for use as encoder pins
  revolverEncoder.attachHalfQuad(33, 32);
  revolverEncoder.setCount(0);
}

// ##########################################################################
// ################    MAIN    ##############################################
// ##########################################################################

void loop() {
  // put your main code here, to run repeatedly:
  printStatement();
  switch (state) {

    case 1:  // initial and waiting for trigger pull
      moveRevolver45degrees();
      treatTrigger();
      if( (triggerFlag) || (CheckForButtonPress()) ) {
        triggerResponse();
        ButtonResponse();
      }
      break;

    case 2:
      moveRevolver45degrees();
      break;

    case 3:
      solActuation();
```

```
      moveRevolver45degrees();
      break;
    }
}


// ++++++++++++++++++++++++++ FUNCTIONS ++++++++++++++++++++++++++++
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

bool CheckForButtonPress() {
  return buttonIsPressed && debounceFlag;
}


void ButtonResponse(){
  buttonIsPressed= false;
  state = 2;
}

// Function to be called when trigger is is activated
// 1. Stops top motor from moving
// 2. Starts timer countdown to activate solenoid
// 3. Moves into states 2
void triggerResponse() {
  timerStart(treatTimer);
  desPOS  += inc45 ;
  triggerFlag = false ;
  state = 2;
  stopTop();
  timerStop(topTimer);
}


// Function for handling duration of solenoid operation
void solActuation() {
  currentTime = millis() ;
  elapsedTime = currentTime - lastCycleTime ;
  if (elapsedTime > cycleDuration) {
    digitalWrite(SOL,LOW);
    state = 1;
    timerStart(topTimer);
  }
}


// Function for sensing treat pull
void treatTrigger() {
  if (scale.is_ready()){
    reading = scale.read();
    if ( reading <= -60000) {
      triggerFlag = true;
    }
  }
}
```

```
// Call this function to stop the top motor
void stopTop() {
  digitalWrite(AIN_1, LOW);
  digitalWrite(AIN_2, LOW);
}


// Function to get revolver to move 45 degrees
void moveRevolver45degrees() {
  currPOS = revolverEncoder.getCount();
  error = desPOS - currPOS ;
  e_sum = e_sum + error;
  P = Kp*error/10;
  I = Ki*e_sum/100;
  D_revolve = P + I;

  if (D_revolve > (MAX_SNACK_MOTOR_VOLTAGE)) {
    D_revolve = MAX_SNACK_MOTOR_VOLTAGE;
    } else if (D_revolve < -MAX_SNACK_MOTOR_VOLTAGE) {
    D_revolve = -MAX_SNACK_MOTOR_VOLTAGE;
  }


  if (D_revolve > 0) {
    ledcWrite(BIN_1, LOW);
    ledcWrite(BIN_2,  D_revolve);
  } else if (D_revolve < 0) {
    ledcWrite(BIN_2, LOW);
    ledcWrite(BIN_1, -D_revolve);
  } else {
    ledcWrite(BIN_2, LOW);
    ledcWrite(BIN_1, LOW);
  }
}
// Function to print out critical values for debugging
void printStatement() {
  Serial.print("State: ");
  Serial.print(state);
  Serial.print(" | Reading: ");
  Serial.print(reading);
  Serial.print(" | Direction: ");
  Serial.print(motorDirection);
  Serial.print(" | Des Pos ");
  Serial.print(desPOS);
  Serial.print(" | Cur Pos: ");
  Serial.print(currPOS);
  Serial.print(" | Revolver PWM: ");
  Serial.print(D_revolve);
  Serial.print(" | I term: ");
  Serial.println(I);
}
```

Appendix D : Circuit and State Diagram