Group 15: Jesse, Miguel, Conor

I. Opportunity

Our goal for this project was to create an interactive product that would help inform and educate younger generations about problems of wealth inequality that affect the United States specifically. In the United States, the top 10% of households hold roughly 70% of the wealth while the bottom 50% hold only 2%. To demonstrate the cyclical nature of poverty, we chose the form of a claw machine to attract younger audiences and hoped to give them an enjoyable experience while leaving them with something to ponder moving forward.

II. Strategy

We created a claw machine that extended further based on how much money was added to it. More expensive, luxury items were placed at the bottom while basic necessities were placed at the top, but aside from this twist the machine operated exactly like a normal claw machine. Initially we had wanted the gantry to reach all four corners of the framed claw machine but due to limitations in the belt we purchased and some spacing issues we could only reach about 80% of the total region we initially desired. Additionally we had wanted to include a slanted or stair like structure to highlight the nature of more expensive prizes at the bottom but pivoted to simply stacking prize boxes on top of each other.

III. Integrated Device

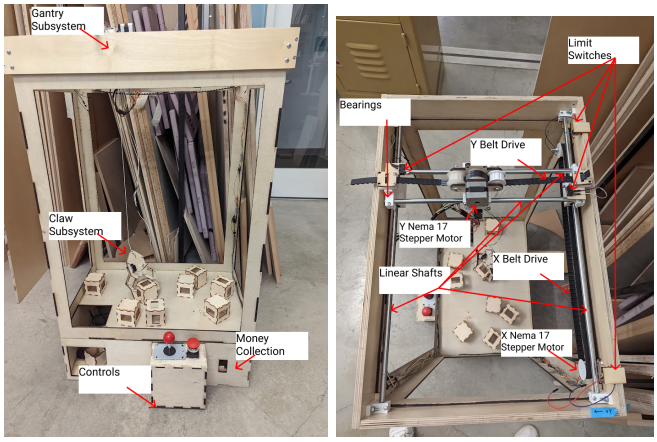

Figure 1,2: Claw Machine, Gantry Subsystem

The gantry utilizes 2 Nema 17 Stepper motors which were each responsible for moving a belt transmission in the X and Y direction respectively. Additionally, limit switches were in place to prevent the gantry from running into the walls of the frame. Bearings and shafts were purchased but pulleys and brackets were 3D printed in an effort to save cost.

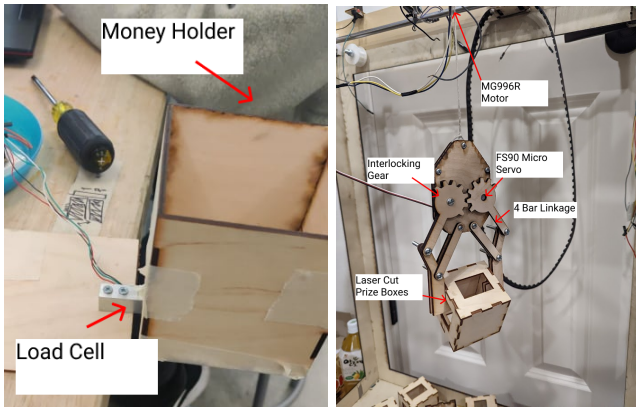

Figure 3,4: Money Collection Subsystem & Claw Subsystem

The coin collection operated off of a laser cut box which accepted quarters and an HX711 Load Cell was connected to an amplifier which dictated how many quarters were present. The Claw had a FS90 Micro Servo responsible for actuating an interlocking gear and four bar system that opened and closed the grippers. Additionally, there was a MG996R servo responsible for dropping and lifting the claw and prize.

IV. Function Critical Decisions

Since we used 4 motors, most of our function critical calculations revolved around these and the calculations are listed below.

Motor 1	Purpose: Lift claw + payload up and down	
Payload Weight	35.7g	weight of a printed payload box
Claw Weight	235 g	weight of claw with Motor 4 + screws
Total Load	300 g	Suggest we Round up for Factor of Safety
Motor Pulley Dia	2.54 cm	Room allowed within Gantry
Required Torque	**762 kg-cm**	
MG996R Servo Torque	2.5kg-cm	Motor 1 able to hold its expected load

The MG996R is a high torque yet lightweight motor which allows us to easily lift and lower the claw with no fear of an item falling or breaking. For the NEMA motors, the Y motor will be used for critical calculations only as it must carry the same weight as the X motor plus the additional weight of the Y belts and bearings.

Motor 2/3	Purpose: Accentuate X-Y Gantry	
Weight of Belts	98 grams	Upon Measurement
Weight of Pulleys	136 grams	Upon Measurement
Miscellaneous Weights (Bushings, washers etc.)	50 grams	Estimation
Coefficient of friction	0.1	Found for factor between stainless steels
Total Load	1.02 kg	Rounded up for Safety
Motor Pulley Dia	2.54 cm	Size of Pulleys created
Required Torque	**2.59 kg-cm**	
NEMA 17 Hold Torque	3.2 kg-cm	Motor 2/3 able to hold their expected weights

The final motor which would be a part of the claw itself was one which we wanted to be small and lightweight. The motor would also have to be relatively durable as it would be potentially moving around the most and the most likely to get damaged.

Motor 4	Purpose: Actuate Claw	
Payload Weight	35.7 grams	Upon Measurement
Coefficient of Friction of Rubber on Plywood	0.95	
Gripping Force Required	37.75 grams	(Prize Box Weight)/Coeff Friction
Moment arm for finger	10.14 cm	
Required Torque	**.374 kg-cm**	

FS5103R Hold Torque	3.2 kg-cm	Motor 4 able to hold expected weight

Other critical decisions involved which belts, shafts, bearings and other gantry parts to use. Most of these decisions came down to price and sizing. We decided that some pulleys would be too expensive and thought that PLA would suffice and be well worth the money saved.

V. Circuitry and State Diagrams

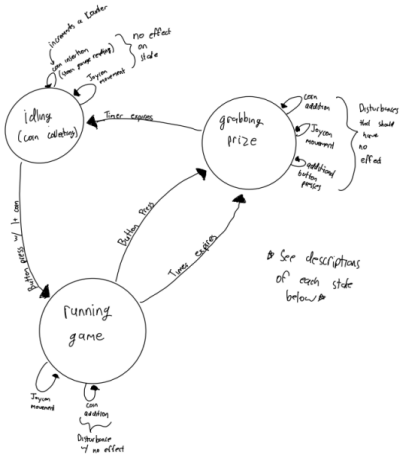

Figure 6: State Diagram and **Figure 7: Circuit Diagram**

Our machine worked on a 3 state system. First, there was an idle state where the game collected money and waited for a button press to enter the game state. Next, in the game state the machine accepted joystick inputs to move the gantry left and right. During this time, it accepted a button press that would initiate the next game state, the prize collection process. The prize collection state involved running a series of commands to: lower the claw a set distance based on the number of quarters, close the gripper, return the claw to its default state, navigate the gantry to the 0,0 position over the prize chute, and finally open the gripper, releasing the prize. From here the machine's state will reset to idle.

VI. Reflection

We think that having a stable CAD model before construction was incredibly beneficial for our work and helped us quickly create the gantry. If we could redo the project, we think that starting before Thanksgiving break or working over break would be beneficial due to the amount of man hours the project took. Additionally, we would recommend spending a larger amount of money on pulleys as these could have run slightly smoother. We also would have spent slightly more time measuring out exact distances for belt drives or using tensioners to ensure tension was present throughout the drive. Overall, this is a project that we hope to pursue further and ultimately perfect.

Appendix A: Bill of Materials

Item	Purpose	Product Details	Count	Price Per	Total Price	Source/Link
Buttons	Stops Claw from moving	600V 6A 38 mm 1.5" Red Sign Momentary Mushroom Push Button Switch	1	7.49	7.49	Amazon
Rope	Lower and Raise Claw	PP Rope, 1/8" diameter, 1 ft	1	0.05		
Bushings	Idlers for Timing Belt	1/4" shaft dia flanged nylon bushings	6	0.69		McMaster
Frame Wood	Create the Frame	1/4" Plywood	2	13.32	26.64	Jacobs
Hardware	Washers/Nuts	M4 washers, M4 hex nuts	15	0	0	Jacobs Provided
Joystick	Actuate	5 Pin Stick 5P Rocker 4 - 8 Ways Joystick	1	11	11	Amazon
Linear Rails	Movement along motor	2 pack linear motion rod shafts and linear bearings	1	21.99	21.99	Amazon
Load Cell	Weight Sensor	Digital Load Cell Weight Sensor 1KG + HX711	1	0	0	Micro Kit
Prize Boxes	Hold Prizes	1/8" x 24" x 48"	1	8.88	8.88	Jacobs
Processor	ESP32	Microcontroller	3	0	0	Micro Kit
Screws	Hold Acrylic to Plywood, Hold Plywood Together	**M4 Socket Head Screws**	25	0	0	Jacobs Provided
Stepper Motor	Open/Close Claw	Stepper motor - FITEC micro servo	1	0	0	Jacobs
Servo Motor	Raise and Lower Claw	MG996R Motor	1	0	0	Owned prior
Stepper Motors	Pulley System	Stepper motor - NEMA-17 size - 200 steps/rev, 12V 350mA	2	15.54	31.08	Jacobs
Timing Belt	For X/Y motion	L Series Timing Belt, 1/2" width	2	11.12	22.24	McMaster

Wire Housings	Keep Cables Organized		1	0		Jacobs
TOTAL					129.32	

Appendix B: Images of CAD

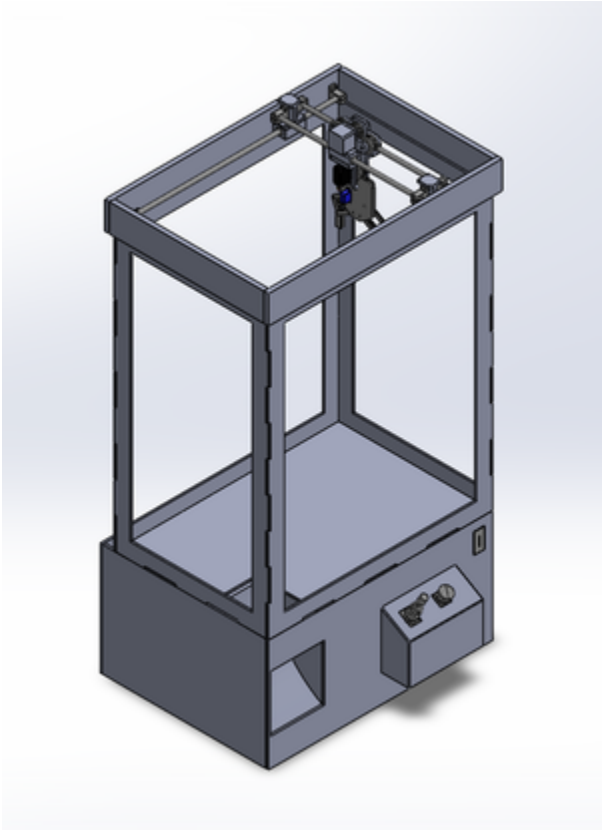

Figure 1: Fully Assembled Design

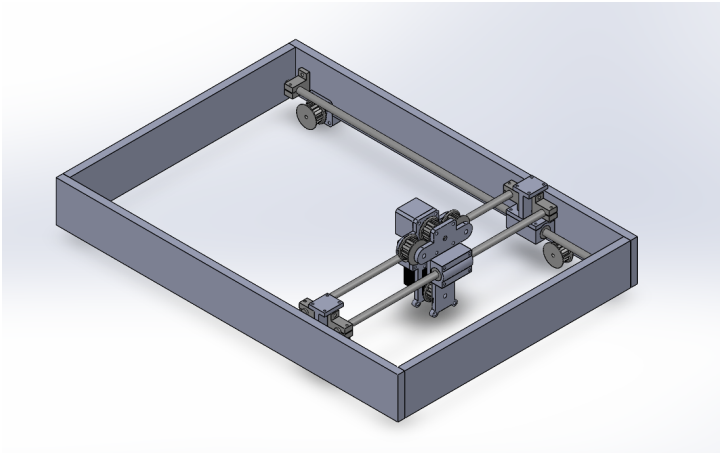

Figure 2: Gantry Transmission Subsystem

Appendix C: Code

submissionClaw §

```
#include <Servo.h>
#include <Stepper.h>

//Joystick Pinout
#define posX 32
#define negX 23
#define posY 22
#define negY 14

//Motor Pinout
#define YdirPin 27
#define YstepPin 33
#define XdirPin 13
#define XstepPin 12
#define microstepPin 15
#define servoPin 18 //Claw Servo
#define zservoPin 5 //Z motion servo

Servo claw;
Servo zmot;

//Sensor Pinout
#define limPosX 19
#define limNegX 16
#define limPosY 17
#define limNegY 21
#define button 4

//VARIABLES
#define stepperDelay 1000
#define stepsPerRevolution 100
#define x 0       //variables to contain coordinates of gantry head
#define y 0


int NumberQuarters = 5;
int pos = 0;
volatile bool stepperLock = false;
volatile bool buttonPress = false;
volatile bool timerInterrupt = false;
int gameState  = 0;
int pressCount = 0;
String pressed = "PRESSED ";
String times = " TIMES";
hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
```

```
//------------------------- BUTTON FUNCTIONS -------------------------------------
void IRAM_ATTR isr(){
  Serial.println("Press Registered");
  if(!timerInterrupt){
     Serial.println("Timer Started");
     buttonPress = true;
     timerRestart(timer);
  }
}

void IRAM_ATTR onTime() {
  portENTER_CRITICAL_ISR(&timerMux);
  timerInterrupt = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux);
  timerStop(timer);
  Serial.println("Timer Stopped");
}

void IRAM_ATTR TimerInterruptInit() {  //The timer simply counts the number of Tic generated by the quartz. With a quartz clocked at 80MHz, we will have 80,000,000 Tics.
  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, 150000, true);          // sets how many tics will you count to trigger the interrupt
  timerAlarmEnable(timer); // Enables timer
  timerStop(timer);
}

void IRAM_ATTR successfulPress(){
  pressCount++;
  Serial.println(pressed + pressCount + times);
  buttonPress = false;
}

//---------------------- MOTOR FUNCTIONS ---------------------------------------


void drop(){
  zmot.attach(zservoPin);
  zmot.write(0);
  for(int posDegrees = 0; posDegrees <= 180; posDegrees++) {
       zmot.write(posDegrees);
       delay(20);
    }
    zmot.write(180);
    delay(4500);
  zmot.detach();
}


void lift(){
   zmot.attach(zservoPin); //Attaches zmotor Pin
   for(int posDegrees = end_angle; posDegrees >= start_angle; posDegrees--) {
         zmot.write(posDegrees);
         delay(20);
     }
     zmot.write(-180);
     delay(7500);
   zmot.detach();
}

void grab(){
  claw.attach(servoPin);
  int start_angle = 50;
  int end_angle = 0;
  for (pos = start_angle; pos >= end_angle; pos--){
    claw.write(pos);
    delay(40);
  }
  claw.write(0);
  delay(1500);
  claw.detach();
}

void releasePrize(){
  claw.attach(servoPin);
  int start_angle = 0;
  int end_angle = 50;
  for (pos = start_angle; pos <= end_angle; pos++){
    claw.write(pos);
    delay(40);
  }
  claw.write(50);
  delay(1500);
  claw.detach();
}

void rotateStepper(int stepPin){
  for (int i = 0; i < stepsPerRevolution; i++) {
    // These four lines result in 1 step:
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(stepperDelay);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(stepperDelay);
  }
}
```

```
void DriveYMotorPositive(){
  digitalWrite(YdirPin, LOW);
  rotateStepper(YstepPin);
  }
void DriveYMotorNegative(){
  digitalWrite(YdirPin, HIGH);
  rotateStepper(YstepPin);
  }
void DriveXMotorPositive(){
  digitalWrite(XdirPin, HIGH);
  rotateStepper(XstepPin);
  }
void DriveXMotorNegative(){
  digitalWrite(XdirPin, LOW);
  rotateStepper(XstepPin);
  }


void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  //Setup Motors
  pinMode(YdirPin, OUTPUT);
  pinMode(XdirPin, OUTPUT);
  pinMode(YstepPin, OUTPUT);
  pinMode(XstepPin, OUTPUT);
  pinMode(servoPin, OUTPUT); //clawServo
  pinMode(zservoPin, OUTPUT); //Zmotor
  pinMode(microstepPin, OUTPUT);
  digitalWrite(microstepPin, HIGH);
  claw.attach(servoPin);
  claw.write(50);
  delay(1000);
  claw.detach();

  //Setup Joystick
  pinMode(posX, OUTPUT); digitalWrite(posX,LOW);
  pinMode(posY, OUTPUT); digitalWrite(posY,LOW);
  pinMode(negX, OUTPUT); digitalWrite(negX,LOW);
  pinMode(negY, OUTPUT); digitalWrite(negY,LOW);



  //Setup Sensors
  pinMode(limPosX, INPUT_PULLUP);
  pinMode(limPosY, INPUT_PULLUP);
  pinMode(limNegX, INPUT_PULLUP);
  pinMode(limNegY, INPUT_PULLUP);
  pinMode(button, INPUT_PULLUP); //button
  attachInterrupt(button, isr, RISING);

  TimerInterruptInit();
}
```

```
void loop() {
  switch (gameState){
    case 0:                                 //Inserting Money
      //Check for Button Press
      if(buttonPress && timerInterrupt){
        successfulPress();
        portENTER_CRITICAL(&timerMux);
        timerInterrupt = false;
        portEXIT_CRITICAL(&timerMux);
      //Check for coins inserted
      readStrainGauge();
      }
    case 1:                                 //Playing the Game
      //Check for Button Press
      if(buttonPress && timerInterrupt){
        successfulPress();
        portENTER_CRITICAL(&timerMux);
        timerInterrupt = false;
        portEXIT_CRITICAL(&timerMux);

      //Check for Limit Switch
       if (digitalRead(limPosX) == LOW){
          Serial.println("RIGHT LIMIT");
       }
       if (digitalRead(limPosY) == LOW){
          Serial.println("UPPER LIMIT");
       }
       if (digitalRead(limNegX) == LOW){
          Serial.println("LEFT LIMIT");
       }
       if (digitalRead(limNegY) == LOW){
          Serial.println("LOWER LIMIT");
       }

      //Check for Joystick Press
       if (digitalRead(posY) == HIGH && digitalRead(limPosY) == HIGH && stepperLock == false) { //and limit switch is low
           Serial.println("up"); //make motor go
           DriveYMotorPositive();
       }
       if (digitalRead(posX) == HIGH && digitalRead(limPosX) == HIGH && stepperLock == false) {
           Serial.println("Right");
           DriveXMotorPositive();
       }
      if (digitalRead(negY) == HIGH && digitalRead(limNegY) == HIGH && stepperLock == false) {
        Serial.println("down");
        DriveYMotorNegative();
        }
      if (digitalRead(negX) == HIGH && digitalRead(limNegX) == HIGH && stepperLock == false) {
        Serial.println("Left");
        DriveXMotorNegative();
        }
      }
    case 2: //Collecting Prize
      stepperLock = true;
      drop();
      grab();
      lift();
      zeroClaw();
      releasePrize();
      stepperLock = false;
      resetMachine();
      gameState = 0;
  }
}
```