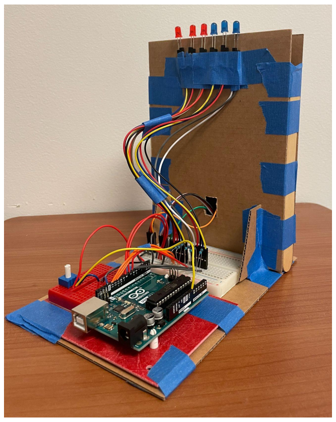
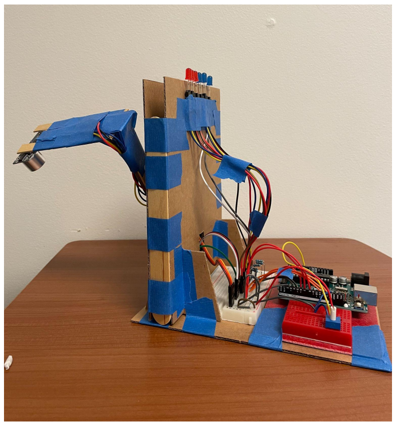
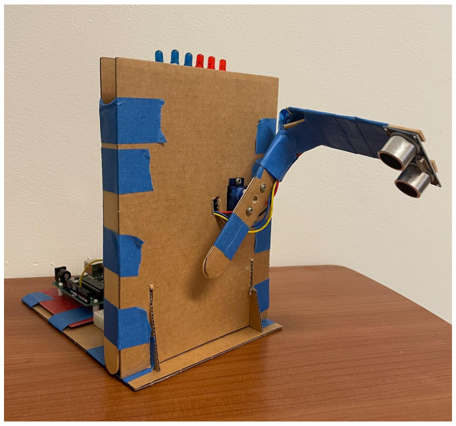
Kate Mosle (km662)
MAE 3780 Individual Project
Final Report
Fall 2021

SlapJack Jr.

I. OVERVIEW

SlapJack Jr is a single-player version of a children's game where one player extends their hands forward, roughly at arm's length, with their palms down. The other player's hands, also roughly at arm's length, are placed, palms up, under the first player's hands. The object of the game is for the second player to flip their own hands over the opposing player's and slap the back of their opponent's hands before they can pull them away. If the slapping player is able to hit their opponents hands, then they earn a point. If the slapping player misses, then their opponent earns a point.

The computer is only able to play as the slapping player, and has a single actuating arm, controlled by a positional servo motor and equipped with an HC-SR04 ultrasonic distance sensor in the 'hand'. The distance sensor detects whether the player's hands are in place or if they have been pulled away. The arm is fixed at a 90 degree angle and has a 180 degree rotational range. There are 3 LEDs of each color at the top of the front panel to denote the points earned by each player. The computer will run LED chase programs when the player wins or reaches 'game over', at which point the scores will reset and the player can continue playing. The potentiometer allows the player to set the difficulty of the game, which is determined by the speed at which the arm moves and which strategies the computer uses. The two strategies available are 'feint', where the arm moves to a random angle between 0 and 100 degrees, or 'full turn', where the arm will move the full 180 degrees to try to score a point. The computer will only make a play from the 0 degree position if the player's hand is in place over the sensor.

II. DESIGN CONSIDERATIONS

Given more time, I would add a 3D printed cover for the potentiometer, so that users do not have to use the screwdriver. The LEDs could have been attached to the other side of the back panel such that the wires were hidden between the front and back panel, with the wires coming out the hole of the back panel to connect to the breadboard, for a cleaner final design. Future design modifications that would also fit within the given budget include the addition of a second arm. A single servo was able to handle the actuation of the arm without any additional mechanical advantage, so a second arm could be added with another ultrasonic sensor with some calculation to ensure the servo is not overloaded. A second ultrasonic sensor was already purchased for this iteration of the design, so the extra cost would only be $1.93 for another 4-wire F-F harness and M-M header as there is enough cardboard and popsicle sticks in this bill of materials for a second arm. The arms could be controlled independently through the use of the positional and the 360 continuous rotation servos, which would not add to the budget cost, but would require modification of the Arduino code. For a more refined final product, I would consider using laser cut acrylic plates rather than the cardboard for the panels, base and supports. However, the additional cost of the acrylic and RPL manufacturing did not make this an option with the current budget constraint. I would also explore using a different, smaller proximity sensor, as the HC-SR04 is quite heavy. One potential option would be to use IR sensing, which would likely fit within the current budget but appropriate parts would need to be sourced. Aesthetic modifications within the current budget could include the use of stronger tape which would allow the builder to use less while still ensuring a secure structure. I would also recommend soldering as many components together as possible as most issues that arose during the build were due to loose connections. The guidelines for this project did not allow for major modifications so no soldering was done in this version.

III. ASSEMBLY INSTRUCTIONS

1. Fix the Arduino board and large breadboard to an unaltered piece of cardboard using tape. The USB port on the Arduino should be at one edge of the panel and parallel with the long dimension. The breadboard fit best when the center channel was aligned with the short dimension of the panel (refer to Figure A6).
2. Place potentiometer on mini breadboard (see Figure A6)
3. Use Figures A2a, A2b and A2C from the appendix and box cutter to cut the cardboard for the front and back panels and 2 supports. Make the slits in the bottom of the panels approximately 1/16” wide in order to fit snuggly with the supports.

4. Insert supports into slots on the back panel and fix with tape.
5. Secure the assembled panel to the front edge of the base (in front of the breadboard) from step 1 using tape.
6. Wire LEDs using 4-wire harnesses and 3-pin headers; connect to the large breadboard. Align LEDs along top of back panel and fix with tape so they are visible over the top of the panel (see Figure A6)
7. Attach the servo to the front panel using included screws. Add a straight bar attachment to the servo and secure it with the included screw (see Figure A4).
8. Use Figure A2d and scissors to cut 3 popsicle sticks to appropriate dimensions and make holes. Assemble the 2 sticks with cutouts to form a window and secure the ultrasonic sensor into the window so it is flush, secure with tape.
9. Place the popsicle sticks at 90 degrees and secure at the angle with tape (see Figure A6).
10. Secure single popsicle stick to servo attachment using 2 included screws through small premade holes in popsicle stick.
11. Connect a 4-wire harness to the ultrasonic sensor and thread wires through the small square hole on the front panel. Secure wires to popsicle sticks fixed to the servo using tape, leaving some slack to ensure wires do not disconnect.
12. Thread sensor harness and servo wires through hole in back panel and wire to large breadboard using headers.
13. Connect the front panel to the back panel using 2 popsicle sticks as spacers and align slots of the front panel over supports. Secure with tape.
14. Complete wiring using circuit diagram in Figure A1. LEDs connected to pins 4 and 11 should be in the center and LEDs connected to pins 6 and 13 should be on the outside of the LED setup. See additional diagram in Figure A5.
15. Assembly the 6V battery pack and connect it to the large breadboard. Connect the Arduino to a computer using the USB cable and download the Arduino code from the Appendix.

Note 1: reference figures for assembly located in Appendix

Note 2: final assembly modifications may be needed to align the popsicle stick attached to the servo in the vertical position with the sensor facing up when the device is first turned on (the servo will move to the 0 degree position). Modify by using a screwdriver to loosen the screw securing the straight bar attachment to the servo, rotate appropriately and reattach (accessible through large hole in the middle of the single popsicle stick of the arm).

IV. OPERATIONAL INSTRUCTIONS

After the device is assembled according to the instructions listed above, use the screwdriver to turn the potentiometer to the fully clockwise position to make sure the game is paused. The Arduino code establishes 4 equal zones on the potentiometer: pause, easy, medium, and hard. Once the player is ready, the potentiometer can be set to easy, medium or hard mode by turning the knob counterclockwise until the desired zone is reached. The computer will always wait until the player places their hand within range of the sensor, about 2 inches above (level with the servo). The computer will randomly select between two playing strategies: feint and full turn. Hard mode will have a higher probability of playing a feint and move with a higher angular speed, and easy mode will be more likely to play a full turn and move with a slower angular speed. On a feint, the servo will rotate to a random angle between 0 and 100 degrees, before returning to the 0 degree position. The player cannot score a point with the strategy, but if they pull their hand away from the sensor then the computer will get a point. On a full turn play, the servo will turn a complete 180 degrees before returning to the 0 position. The computer will score a point if the turn is completed before the player can pull their hand away. If the player can pull their hand away before the turn is complete, then they will earn a point. The player must completely remove their hand from above the sensor to be considered ‘pulled away’, and can do so by either moving directly back from the device or to the left. Moving their hand to the right may lead the arm to hit the player’s hand. The game will continue until either the player or the computer is able to achieve 3 points. The LEDs will flash to indicate the winning side, and then the ‘game over’ LED light chase sequence will play. After the sequence is complete, the score will reset to 0 for both the computer and the player, the servo will return to the 0 position, and a new match begins. The player can change the game mode at any point by adjusting the potentiometer.

APPENDIX

A. BILL OF MATERIALS

Table 1. Full list of all materials used for this project. (*) denotes a part included in the kit provided by course staff. (x) denotes parts that were purchased/ordered. (-) denotes parts/quantity that were scavenged.

Item Name	*Vendor*		*Part Number*	*Quant. Ordered*	*Quant .Used*	*Unit Cost*
4-wire harness F-F	Digikey	-	1528-1961-ND	4	4	$0.40
3-pin header M-M	Digikey	* (2) - (4)	3M156516-03-ND	6	6	$0.58
100 k trimpot	Jameco	*	2291079	1	1	$0.97
MicroServo Positional + attachments	DFRobot	*	SER-0006	1	1	$3.30
Arduino Uno	Digikey	*	1050-1024-ND	1	1	$20.90
Wire Kit	Amazon:Austor	*	B07PQKNQ22	1	1	$2.17
Tongue Depressor	In-Stock Supplies	x		10	8	$0.03
Cardboard (5”x7”x1/16”)	In-Stock Supplies	x		10	6	$0.11
LED, red	Jameco	*	697602	3	3	$0.05
LED, blue	Jameco	-	2283655	3	3	$0.05
Resistor, 1 k	Digikey	* (5) - (1)	1.0kQBK-ND	6	6	$0.01
HC-SR04 Ultrasonic Sonar Distance Sensor	DigiKey	x	2234-HC-SR04-ND	2	1	$3.95
Breadboard	Newark	*	79X3922	1	1	$2.71
Mini breadboard	Newark	*	98AC7296	1	1	$1.05

4-AA battery holder	Jameco	*	216187	1	1	$1.75
AA batteries	McMaster-Carr	*	71455K58	4	4	$0.40
USB Cable A to B	Monoprice	*	39918	1	1	$1.09
Small screwdriver	QLP	*	Custom imprint	1	1	$ -
Blue Painters Tape		-		1	1	$ -
Box cutter		-				$ -
Scissors		-				$ -

Total Cost from Scratch:	$50.28
Total Budget (purchased/scavenged):	$13.23

B. CIRCUIT DIAGRAM

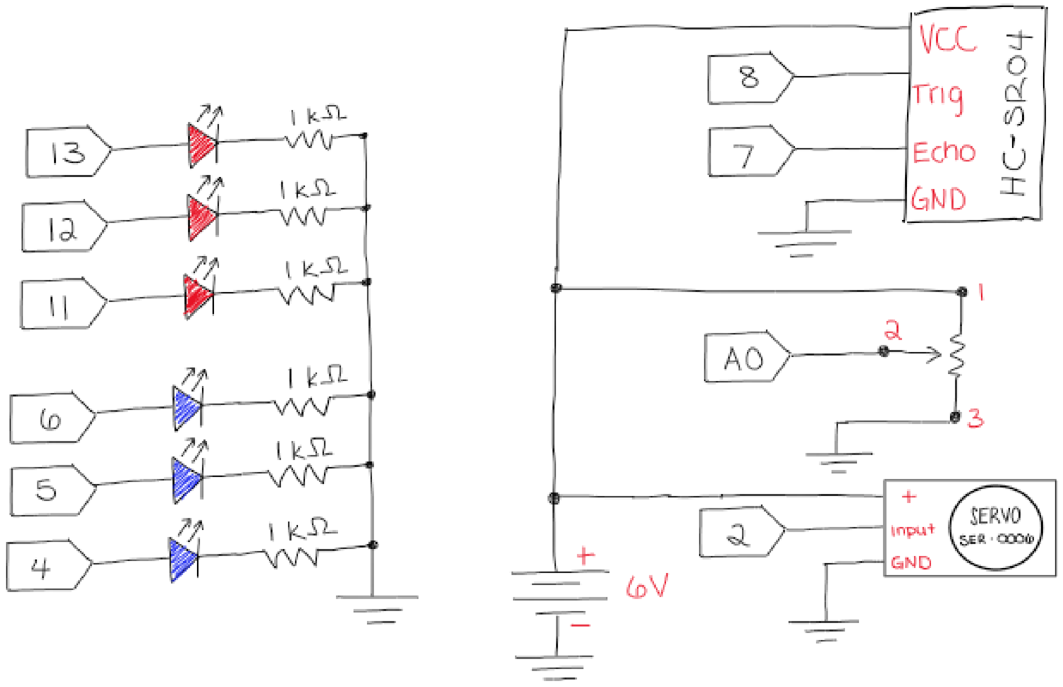
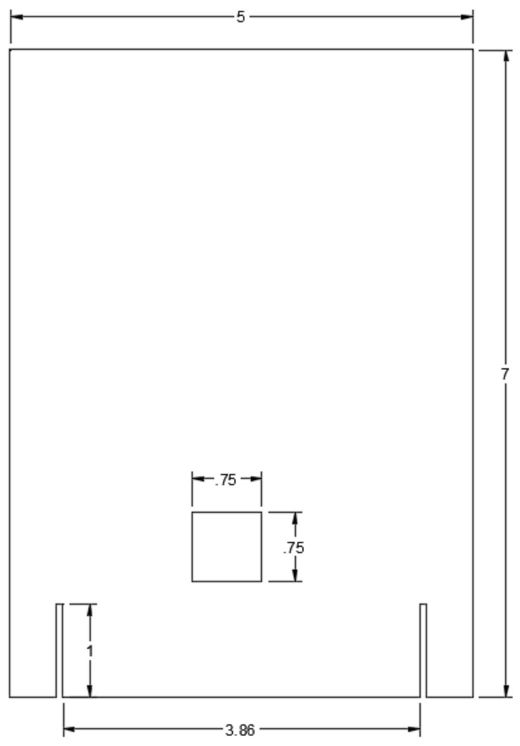
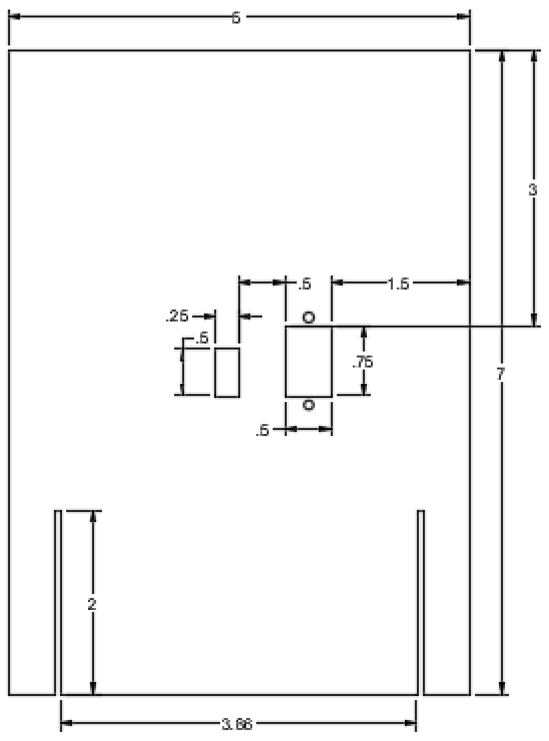


Figure A1. Circuit diagram. All grounds are connected to a single common ground. Pins 4, 5, 6 correspond to blue LEDs. Pins 11, 12, 13 correspond to red LEDs.

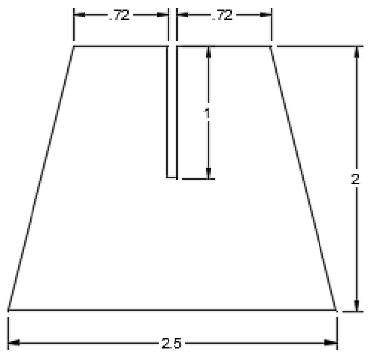
C. CAD FILES & DIAGRAMS



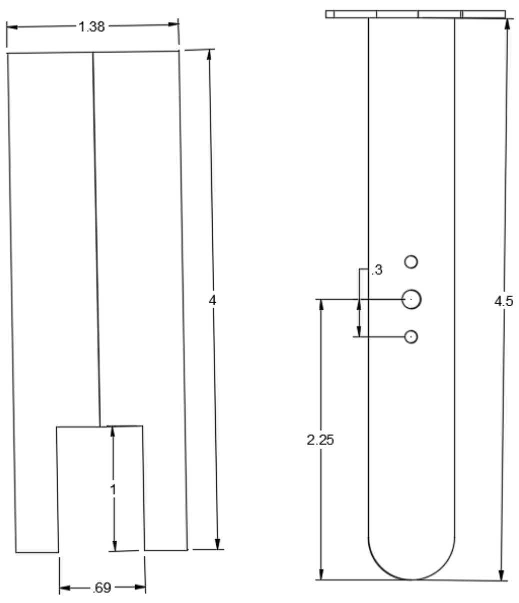
(a)



(b)



(c)



(d)

Figure A2. CAD drawings for (a) back panel, (b) front panel, (c) supports, and (d) arm components. Dimensions are listed in inches.

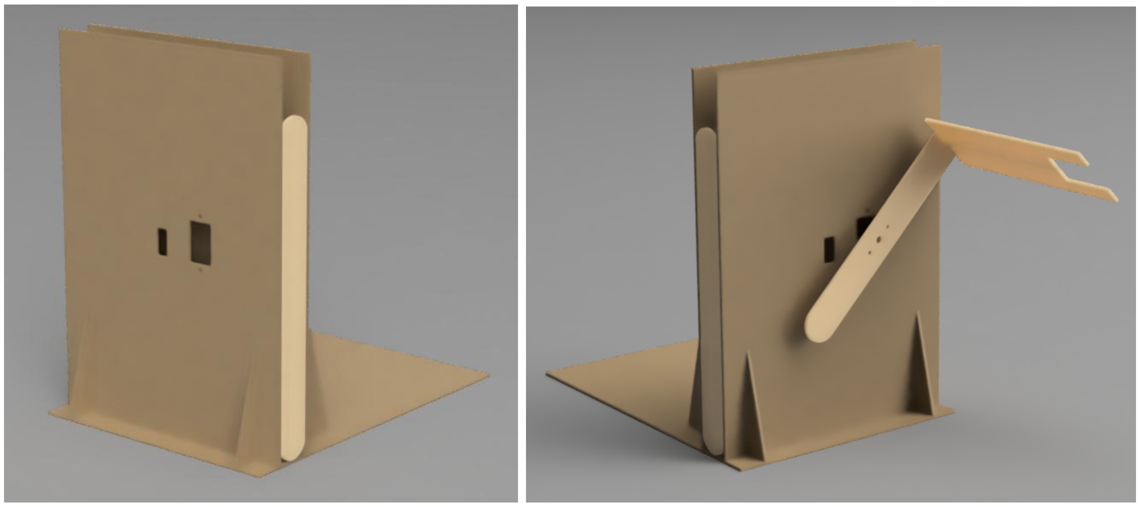


Figure A3. Assembled mechanical structure.

Figure A4. Servo assembly with straight bar attachment.

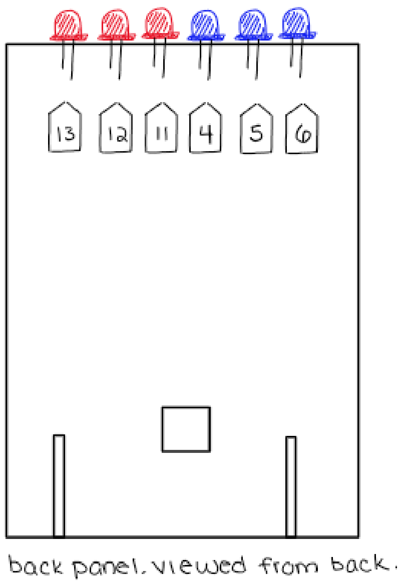
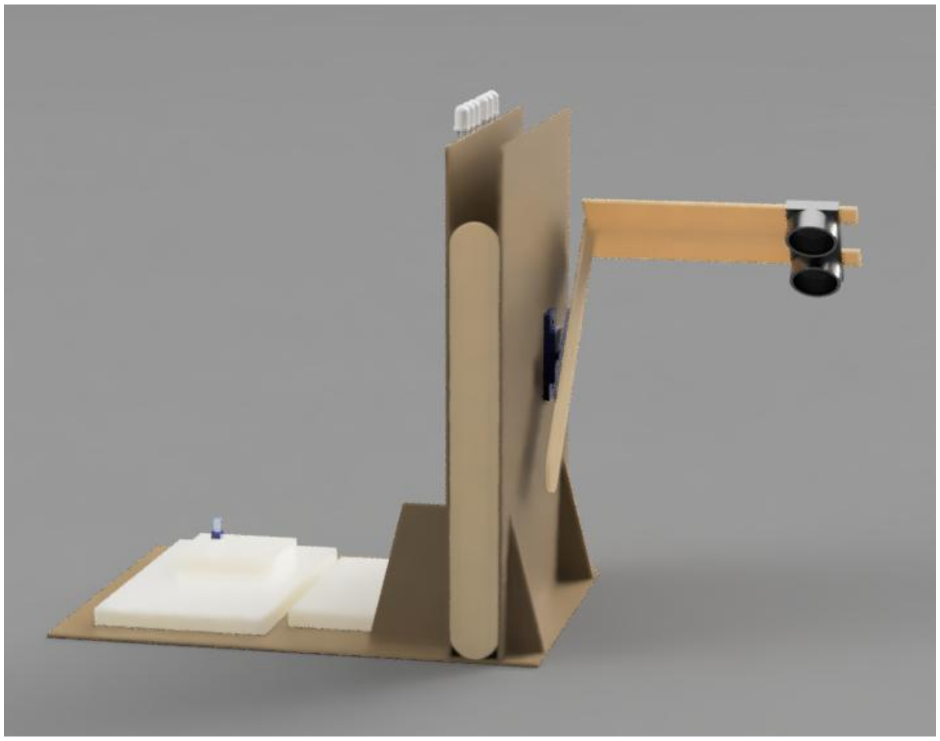


Figure A5. LED-pin diagram..



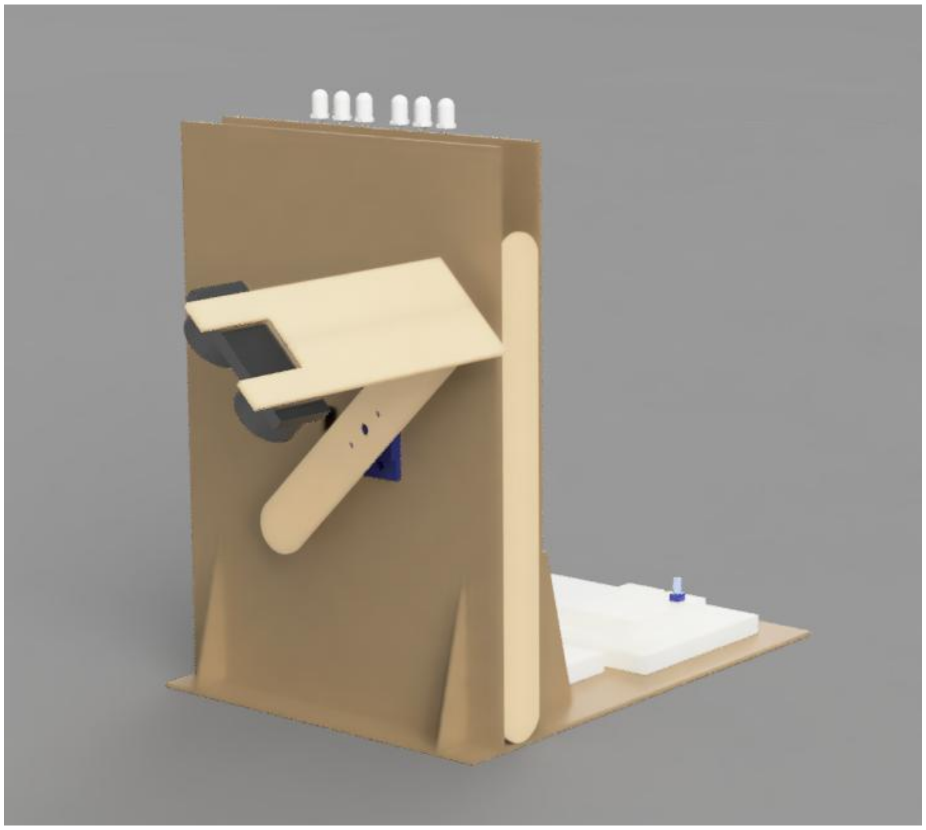


Figure A6. Full assembly with ultrasonic sensor, potentiometer, actuator, LEDs, and arduino & breadboard place markers.

D. ARDUINO CODE

```
//SlapJack

//Goal: beat the player by rotating 180 degrees before the player can pull their hands
away,
// player can earn points by pulling their hands away before the arm makes a full turn.
Computer
// can also earn points by making the player pull their hands away on a feint

//proximity sensor checks if players hands are in place
//game mode determined by potentiometer position (off, easy, medium, hard)
//computer can either feint to a random angle or make a full turn (harder game mode
means computer is more likely to feint)
//computer moves faster at harder game modes

//first to earn 3 points wins, game resets after gameover LED sequence

#include <Servo.h>

Servo servo1;
const int trigger = 8;
const int echo = 7;

float dist;
int speed; //set game mode (speed)
int outofrange = 5; //approx distance between US sensor and player's hands

int computerscore = 0;
int playerscore = 0;

void setup()
{
  Serial.begin(9600);

  //Servo: signal to pin 2, pulse width limits (500,2500)
  servo1.attach(2,500,2500);

  //Distance Sensor: input - 7, output - 8
  pinMode(trigger,OUTPUT);
  pinMode(echo,INPUT);

  //LED (red-computer)
  pinMode(13,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(11,OUTPUT);

  //LED (blue-player)
  pinMode(6,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(4,OUTPUT);

  servo1.write(0); //always start at 0
  Serial.println("setup complete");
}

void loop()
{

  int mode = analogRead(A0); //potentiometer gives number between 0-1023

  //Off
  if (mode < 255)
    speed = 0;
  //Easy Mode
  else if(mode < 511){
    speed = 3;
  }
  //Medium Mode
  else if(mode < 765){
    speed = 2;
  }
  //Hard Mode
  else{
    speed = 1;
  }
  Serial.println(speed);

  //choose strategy based on game mode if player is ready (hard mode more likely to
feint)
```

```
  if (speed!=0 && inRange()){
    Serial.println("play");
    int randstrat = random(11);
    if (randstrat <= speed+3){
      fullturn(speed);
      Serial.println("full");
    }
    else{
      feint(speed);
      Serial.println("feint");
    }
  }
  //time delay before next move (1-2.5 sec)
  delay(500*random(2,6));
}

void feint(int speed){
  //start from 0, turn to final angle at game speed, check for hands, return to 0

  //generate random final angle position between 0 and 100 (incomplete rotation)
  int ran = random(0,100);
  for (int i=5;i<=ran;i=i+5){
    servo1.write(i);
    delay(6*speed);
  }
  //check if player pulled their hands back
  if(!inRange()){
    pointComputer();
  }
  //reset position
  delay(500);
  for (int j=ran-3;j>=0;j=j-3){
    servo1.write(j);
    delay(40);
  }
  servo1.write(0);
}

void fullturn(int speed){
  //start from 0, make full 180 turn at game speed, check for hands, return to 0
  //full turn slightly faster than feint
  for (int i=0;i<=180;i=i+5){
    servo1.write(i);
    delay(3*speed);
  }
  //check for players hands
  if(inRange()){
    pointComputer();
  }
  else{
    pointPlayer();
  }
  //reset position
  delay(500);
  for (int j=180;j>0;j=j-3){
    servo1.write(j);
    delay(40);
  }
  servo1.write(0);
}

bool inRange(){
//checks for the players hands, returns true if their hands are in place
  digitalWrite(trigger,LOW);
  delayMicroseconds(5);
  digitalWrite(trigger,HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger,LOW);
  dist=pulseIn(echo,HIGH); //returns length of pulse in microseconds (dist/58 for cm)
  Serial.println(dist/58);
  return dist/58 <= outofrange;
}

void updateScoring(int p_score,int comp_score){
  //change LEDs to match the current score
  if (comp_score==1){
    digitalWrite(11,HIGH);
    digitalWrite(12,LOW);
    digitalWrite(13,LOW);
  }
  else if (comp_score==2){
    digitalWrite(11,HIGH);
    digitalWrite(12,HIGH);
    digitalWrite(13,LOW);
  }
  else if (comp_score==3){
    digitalWrite(11,HIGH);
    digitalWrite(12,HIGH);
    digitalWrite(13,HIGH);
    gameOver();
  }

  if (p_score==1){
    digitalWrite(4,HIGH);
    digitalWrite(5,LOW);
    digitalWrite(6,LOW);
  }
  else if (p_score==2){
    digitalWrite(4,HIGH);
    digitalWrite(5,HIGH);
    digitalWrite(6,LOW);
  }
  else if (p_score>2){
    digitalWrite(4,HIGH);
    digitalWrite(5,HIGH);
    digitalWrite(6,HIGH);
    gameOver();
  }
}

void pointComputer(){
  //increase computer score by 1, change LEDs
  computerscore++;
  updateScoring(playerscore,computerscore);
}

void pointPlayer(){
  //increase player score by 1, change LEDs
  playerscore++;
  updateScoring(playerscore,computerscore);
}

void gameOver(){
  // play LED game over sequence, reset scores
  digitalWrite(11,HIGH);
  digitalWrite(12,HIGH);
  digitalWrite(13,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  delay(2000);
  digitalWrite(11,LOW);
  digitalWrite(12,LOW);
  digitalWrite(13,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  if (playerscore<computerscore){
    for (int i=1;i<=5;i++){
      digitalWrite(13,HIGH);
      digitalWrite(12,LOW);
      digitalWrite(11,HIGH);
      delay(250);
      digitalWrite(13,LOW);
      digitalWrite(12,HIGH);
      digitalWrite(11,LOW);
      delay(250);
    }
  }
  else{
    for (int i=1;i<=5;i++){
      digitalWrite(6,HIGH);
      digitalWrite(5,LOW);
      digitalWrite(4,HIGH);
      delay(250);
      digitalWrite(6,LOW);
      digitalWrite(5,HIGH);
      digitalWrite(4,LOW);
      delay(250);
    }
  }
  digitalWrite(11,LOW);
  digitalWrite(12,LOW);
  digitalWrite(13,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  for (int i=1;i<=5;i++){
    // light chase victory
    digitalWrite(13,HIGH);
    delay(250);
    digitalWrite(12,HIGH);
    digitalWrite(13,LOW);
    delay(250);
    digitalWrite(11,HIGH);
    digitalWrite(12,LOW);
    delay(250);
    digitalWrite(4,HIGH);
    digitalWrite(11,LOW);
    delay(250);
    digitalWrite(5,HIGH);
    digitalWrite(4,LOW);
    delay(250);
    digitalWrite(6,HIGH);
    digitalWrite(5,LOW);
    delay(250);
    digitalWrite(6,LOW);
  }
  playerscore = 0;
  computerscore = 0;

}
```