Project Contract and Initial Design Document

Names – NetId's – Emails:

Judy Park – jpark379 – jpark379@uic.edu

Julianne Pabona – jpabon3 – jpabon3@uic.edu

Shirley Li – sli235 – sli235@uic.edu

Name of Project:

Gilbert's Fetch Machine

Abstract:

This fetch machine will use multiple I/O devices. This includes (2) LEDs, (2) potentiometer, (1) LCD screen, (1) buzzer, (2) buttons and (2) servos. We will create a fetch machine that will allow pet owners to play fetch with their pets with only a press of a button. The machine will act like a catapult that will fling a ball into the air once the user presses a button. If another button is pressed, the machine will reset the catapult back to its original position. This fetch machine will also consist of two Arduinos which will communicate with each other.

Detailed Project Ideas:

FINAL Overall Description of Project Idea:

Our project's main goal is to create a device that plays fetch with a pet. This is achieved by using a catapult that is programmed to throw a ball when a button is pressed. Once the pet brings the ball back to the catapult, the machine could be returned to its original position when a second button is pressed. There will also be a buzzer to go off and an LCD display that will say "FETCH!" when the catapult goes off and the ball is thrown.

FINAL Project Design stating Expected Inputs/Outputs

Our project will consist of two Arduinos with several input and output devices. Two input devices included in our project are two buttons. One button is in charge of throwing the catapult and the second button is in charge of resetting the catapult to its starting position. Originally, we planned to have two sensors: one to sense the ball on the machine and another to sense when the pet is nearby. However, the sensors were too sensitive and made the catapult go off frequently, even with the slightest amount of movement. A third input device would be a potentiometer to adjust the volume of the buzzer. There will also be several output devices consisting of LEDs, servo motors, an LCD, and a buzzer. One green LED, when lit, will indicate that the ball is thrown. One red LED, when lit, will indicate that the catapult is set to its original position. The LCD will also output the word “FETCH” when the catapult goes off and the ball is thrown. The two servo motors will be attached to a cup that will hold the ball. Once the Arduino receives the input that the pet is nearby, then the two servo motors will catapult the ball in the cup they are attached to. Additionally, when the ball is catapulted, the speaker/buzzer will go off. We will be using two Arduinos--one for the LCD, servos, button and potentiometer connection and another for the LEDs, buzzer, the other button and other potentiometer.

FINAL Plan for Communication

For this project, we are planning to use two Arduinos to communicate to each other through serial communication. In order for them to communicate, we will wire the first Arduino's RX pin to the second Arduino's TX pin and the second Ardiuno's RX pin to the first Arduino's TX pin. Then one button will be wired to each Arduino. When the button wired to the first Arduino gets pressed the buzzer will go off, and it will send that signal to the other Arduino which receives that signal and executes the throw, turns on the green LED, turns off the red LED (if on) and prints fetch to LCD. When the second button is pressed, that sends the signal from the second Arduino to the first Arduino which will reset the catapult, turn on the red LED, turn off the green LED, and reset LCD. The button on each Arduino sends signals to the other Arduino using the serial. When a button is pressed, it will write to serial and the other Arduino will read the serial input and then execute.

FINAL Description of the Original Work

Although there are simpler versions of this project online, we are using six different input and output devices. We are planning to use buttons that will control when the catapult goes off and to reset the catapult to its original starting position. Our plan also consists of a buzzer that will go off when the ball is thrown by the catapult, LEDs that will light up when the ball is thrown and when the catapult is reset, and an LCD screen to display the word "FETCH" when the catapult goes off. The simplified version consists mainly of sensors which senses when there is a ball, but it does not consist of the different input devices we have.

Problems we encountered and how we resolved them

There were many problems we encountered throughout the whole process of creating our project. Our first problem was related to the size of the servos. The servo motors were really small, making it difficult to attach them to anything in order to build the catapult. Because of their small size, it was also difficult to find something that was light enough to put the ball for the servo motors to carry. At first we tried using strips of cardboard attached to the servos to hold up a cup that was used for the catapult. However, that was too heavy for the servo motors. So for our second prototype of the catapult, we made the height a little shorter and used popsicle sticks instead of cardboard. In addition, because of the rounded edges of the cup, the servo motors kept bumping into each other, so we changed the cup into a little box made of cardboard. Another problem we faced with the design of the catapult was that the servo motors had a lot of power when it flung the catapult. So, we had to weigh down the boxes the servo motors were attached to with rocks.

Another problem we encountered was using sensors. Our original design involved using sensors to sense the pet and when the ball was on the catapult. However, after trying to use multiple different types of sensors, the sensors were all too sensitive and would sense any slight movement. Since these sensors were our main way to tell the Arduino to set off the catapult, we tried switching to a serial monitor instead. This method would have also been a method for communicating with an external device. The second problem we faced here was that whenever we tried inputting ‘1’ into the serial monitor for the catapult to go off, it still would not read in the input correctly, since it kept showing as 0. So our final decision was to replace this method with buttons instead. These buttons are used to reset the catapult and to fling the catapult to the throw the ball. This also changed our method for communication, as we had to incorporate a second Arduino.

How to build our project

First, start with building the catapult that will be used to hold the ball. Take cardboard and cut and create a box big enough to hold a ball about 5 inches wide and long. Then cut the cube in half and that will act as the holder for the ball. Then, take a popsicle stick and use a thumbtack to poke through it, about one inch away from the end of the popsicle stick and another thumbtack about half an inch away from the first thumbtack, make sure that the thumbtacks are able to go into the servo attachments holes. Repeat that step with two popsicle sticks. Next, connect servo motors with its attachments and connect the thumbtack on the popsicle sticks with the attachments. One popsicle stick per a servo motor. Use the hot glue gun to glue the ends of the thumbtack. Using the cardboard, create two more cubes about six inches width and height, inside the cube add rocks which will allow the catapult to be held down. Next, use duct tape to tape down the servos one on each box. Then, use duct tape to tape the popsicle sticks to the half cut cube, one popsicle stick on the right side of the cube and other on the left. This will serve as the catapult.

Hardware: First, connect the LCD board and potentiometer onto the breadboard. To connect the LCD screen, you must connect a wire from LCD VSS to GND (the negative side of the breadboard). Next, connect a wire from LCD VDD to 5V (the positive side of the breadboard). Next, connect another wire from LCD VO to the empty spot on the breadboard in between the two legs of the potentiometer on the same bread board.. Additionally, connect LCD RS pin to digital pin 12 on one Arduino. Following that, plug in another wire from the R/W pin to GND (the negative side of the breadboard). Next, connect the LCD Enable pin (E) to digital pin 11. Additionally, connect LCD pins D4, D5, D6, and D7 to the pins 5, 4, 3, and 2 on the same Arduino that RS and R/W are plugged into respectively. Then connect LCD LED+ (A) to 5V through a 220 ohm resistor, meaning that you can plug a 220 ohm resistor anywhere on the breadboard (columns a through j) with the other end of

it plugged into 5V (the positive side of the breadboard) and then you would plug the wire onto the same row as the resistor and make sure it is on the same side on the breadboard. For the last wire for the LCD, plug it from LCD LED- (K) and the other end to GND (the negative side of the breadboard). As for the potentiometer, get one wire and plug it into the immediate right of the side with one leg. Then connect the other end of that wire to the positive side of the breadboard. Next, get another wire and plug it to the immediate left of the side with one leg and connect the other end of the wire to the negative side of the breadboard. Then, in order for the negative side of the breadboard to be plugged to GND and the positive to be connected to 5V, make sure the wires are on the same column respectively. Then, grab another wire and plug it from the positive side of the breadboard to the 5V on the same Arduino as the one plugged in for the LCD. Then, grab a different wire and connect it from the negative side of the breadboard to GND on the same Arduino mentioned before. You must connect the negative side of the breadboard to GND and the positive side of the breadboard to 5V for the other breadboard and Arduino as well.

For the servos, there should be 3 wires attached to each of them. Simply connect the brown wire to the negative side of the breadboard connected to GND that is also used for the LCD screen. Then connect the red wire to the positive side of the breadboard connected to 5V that is also used for the LCD screen. Lastly, connect the yellow wire to Pin 10 on the same Arduino used for the LCD. Repeat the previous steps for the second servo, but instead of connecting the yellow wire to Pin 10, connect that one to Pin 9 on the same Arduino used for the LCD.

Next, we must connect a button. In order to do that, you must place a button in the middle of the breadboard with two legs of the button on column e and the other two legs of the button on column f. Then, assuming that the breadboard is faced upright and vertically, connect one wire onto the same row as the leg on the bottom right to the positive side of the breadboard (columns must be

from g through j). Next, add a wire connected to the same row as the top left leg of the button (with columns from a through d) and Pin 7 of the Arduino. On the top right leg of the button, connect a 10K ohm resistor on the same row on any column that is g through j. Next, on the side of the resistor that is not on the same row as the button, connect a wire that is on the same row as the resistor (columns f through j, minus the column where the resistor is already connected). Then plug the other end of that wire onto the negative side of the breadboard. Add another wire that connects the positive side of one side of the board to the other positive side of the same board along with one other wire that connects the negative side of one side to the other negative side. Repeat these steps for the button on another breadboard and another Arduino.

Furthermore, we must now connect the LEDs. Place one Green LED anywhere on the breadboard with the LCD. then place a red LED on the other breadboard. Grab two 220 ohm resistors and plug the each resistor on the same row as the shorter side of their respective LEDs. You may plug the other end of the resistor on any column of the breadboard, as long as it's on the same side as the LED. Grab another wire and plug it onto the same row as the side of the resistor that is not on the same row as the LED. Then connect the other end of that wire onto the negative side of the breadboard. Next, plug one wire on the same row as the longer side of the LEDs and have the other end of the wire connected to Pin 13 for each LEDs. Do this for each of the LEDs.

For the buzzer and the potentiometer, you may plug the buzzer on the same breadboard as the Red LED. Make sure that you plug in a wire that goes from the black wire of the potentiometer to the negative side of the breadboard (GND). Then, attach the red wire of the potentiometer to the Analog Pin A2. Lastly, attach the yellow wire to the positive side of the breadboard (5V). If you have a potentiometer with legs instead of wires, then attach a wire on the same row as the middle leg to the Analog Pin A2, a wire connected from the same row as the right leg of the potentiometer to

the positive side (5V) of the breadboard and a wire plugged in from the same row as the remaining leg plugged into the negative side of the breadboard (GND). For the buzzer, you must connect a wire that is on the same row as the leg closest to the small '+' on the buzzer. Connect the other end of the wire to pin 6. Next, plug another wire on the same row as the other leg and plug the other end of it onto the negative side of the breadboard (GND).

After connecting the devices, upload the ino files to the Arduinos then connect the two Arduinos by wiring the RX pin 0 on the first Arduino to the TX pin 1 on the second Arduino. Then, do the same for the RX pin 0 on the second Arduino to the TX pin 1 of the first Arduino. After connecting the RX and TX pins, connect the GND pin from the first Arduino to the GND pin of the second Arduino.

How our project is to be used

In order to use this project, place the ball on the catapult. There are two buttons, one for throw and one for reset. After placing the ball on the catapult, press the throw button. Then the catapult will fling the ball for your pet to fetch and on the LCD screen it would print “Fetch!”. The green LED would go on and red LED would go off. In addition, the buzzer will make a noise indicating that the ball is being thrown, you can change the volume by using the potentiometer connected to the buzzer. Once your pet catches the ball, reset the catapult to its original position by pressing the reset button and the red LED would light up. The LCD is also reset. Then, place the ball back onto the catapult and the machine is ready for another round of fetch.

Supporting Materials

Timeline of Development-Completed Items

Date	Plan
10/05 - 10/12	Project Contract
10/12 - 10/19	- Buy/Get all supplies - Start assembling the structural components of machine (cup to hold the ball along with the the components to catapult)
10/19 - 10/26	-Assemble catapult, test and create code to servo
11/02-11/09	-Work on Updated Design Document (Milestone 4) -Test and create code to LEDs, buzzer, lcd screen, sensor and potentiometer
11/09 - 11/16	-Work on code for communication between the arduinos
11/16 - 11/23	-Put different parts of code together for final code file -Work on and submit design presentation
11/23 - 11/30	-Start working on final design document: - Overall description of project - Final plan for communication - Final Project Design stating expected Input/Output
11/30 - 12/7	-Finish and submit final design document

List of Materials Expected to be Needed

- 2 Arduinos
- 3 Breadboards
- Wires
- 2 LEDs
- 2 Servo motors
- 2 potentiometers
- 1 LCD board
- 2 Buttons
- Buzzer
- Cardboard
- 2 Popsicle sticks
- Duct tape
- Hot glue gun
- 4 Thumbtacks
- Rocks
- Ball

List of References(MLA)

"AnalogRead()." *AnalogRead() - Arduino Reference*, www.arduino.cc/reference/en/language/functions/analog-io/analogread/.

"AnalogWrite()." *AnalogWrite() - Arduino Reference*, www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/.

"Make a Simple LED Circuit." *Arduino Project Hub*, create.arduino.cc/projecthub/rowan07/make-a-simple-led-circuit-ce8308.

"Map()." *Map() - Arduino Reference*, www.arduino.cc/reference/en/language/functions/math/map/.

"Serial.write()." *Serial.write() - Arduino Reference*, www.arduino.cc/reference/en/language/functions/communication/serial/write/.

"Serial.read()." *Serial.read() - Arduino Reference*, www.arduino.cc/reference/en/language/functions/communication/serial/read/.

"Servo." *Servo - Arduino Reference*, www.arduino.cc/reference/en/libraries/servo/.

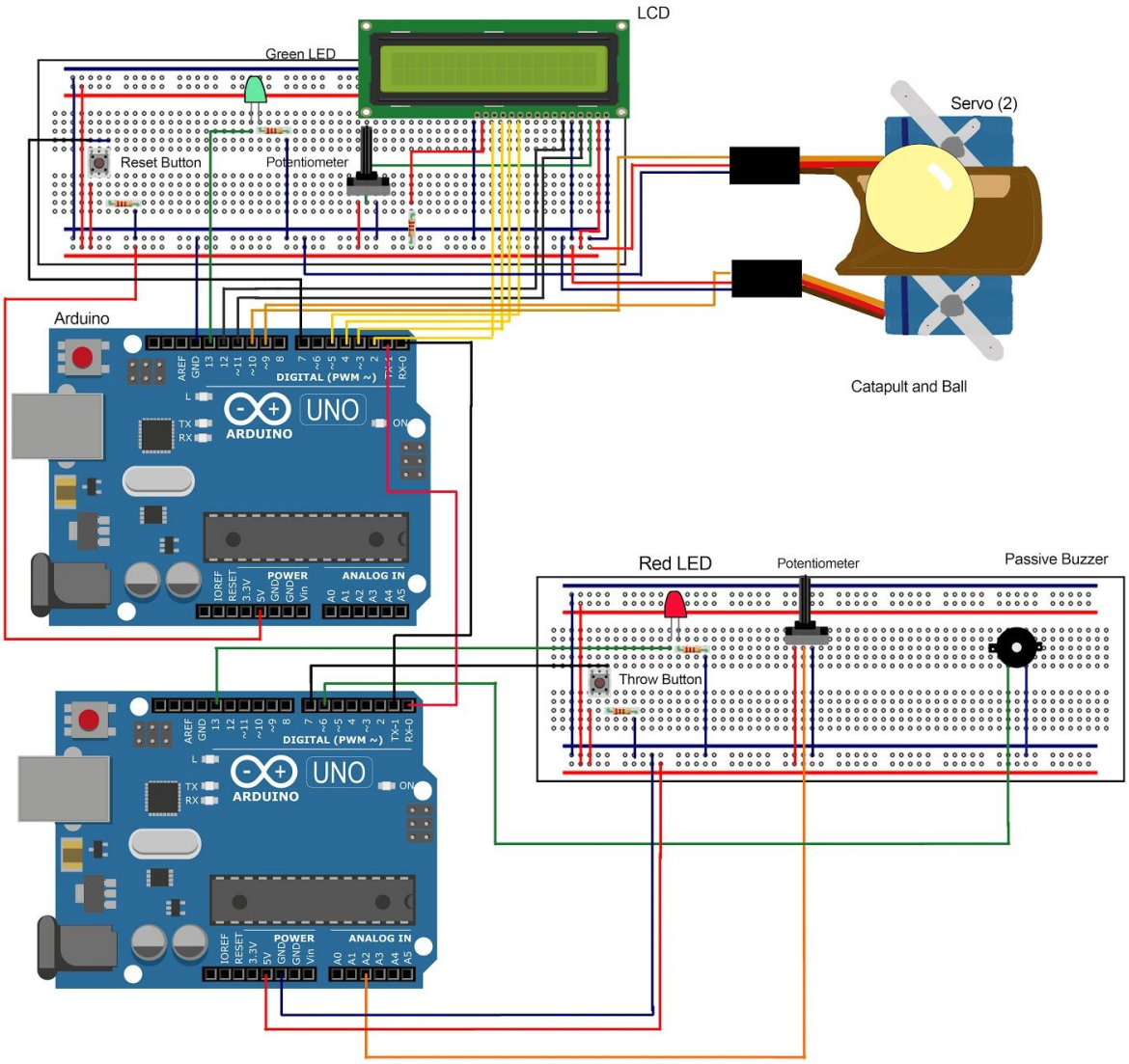
Team, The Arduino. “Button.” *Arduino*, www.arduino.cc/en/Tutorial/BuiltInExamples/Button.

Team, The Arduino. “Debounce.” *Arduino*, www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce.

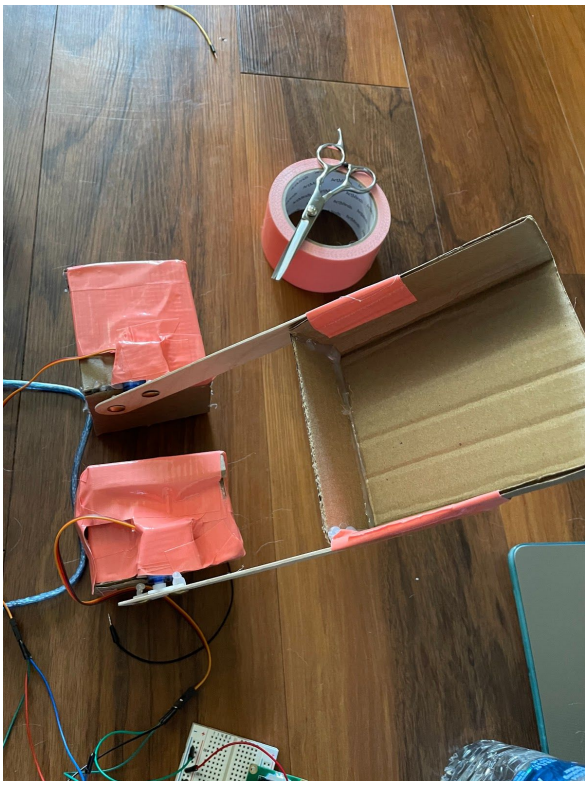
Team, The Arduino. “Display() and NoDisplay() Methods.” *Arduino*, www.arduino.cc/en/Tutorial/LibraryExamples/LiquidCrystalDisplay.

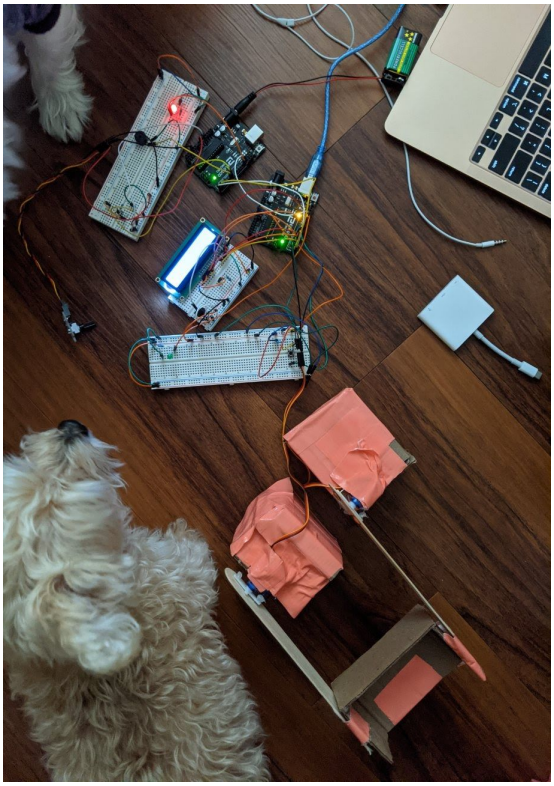
“USE a BUZZER MODULE (PIEZO SPEAKER) USING ARDUINO UNO.” *Arduino Project Hub*, create.arduino.cc/projecthub/SURYATEJA/use-a-buzzer-module-piezo-speaker-using-arduino-uno-89df45.

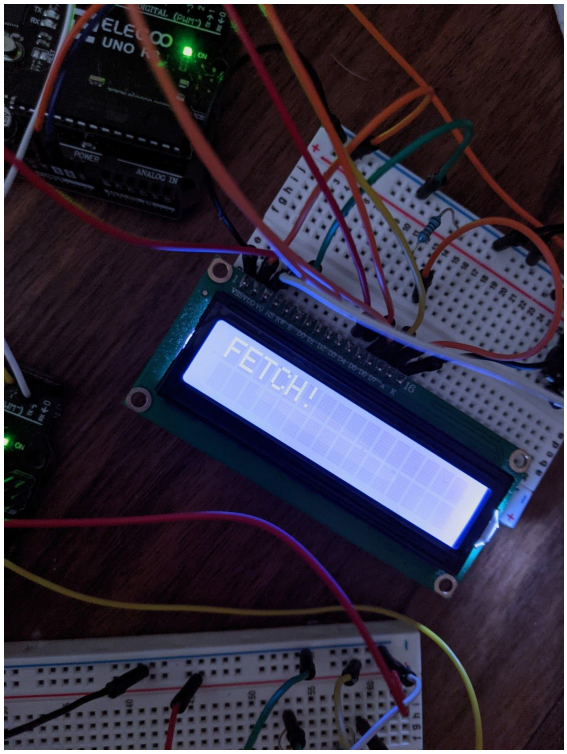
Hardware Diagram

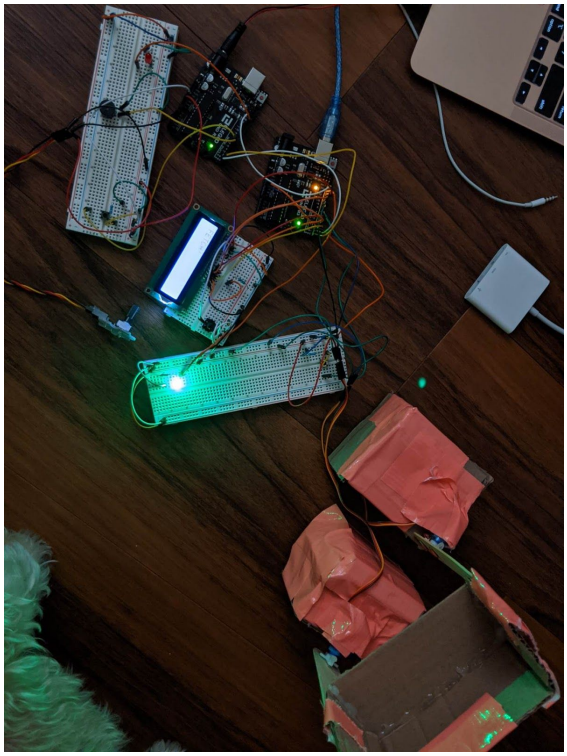


Pictures of Our Project









.ino Code:

SemProj.ino Code:

```
/* semproj.ino */

//
// <JUDY PARK, JULIANNE PABONA, SHIRLEY LI>
// SEMESTER PROJECT
// U. Of Illinois, Chicago
// CS 362: Fall 2020
//

/* Description: Code that uses two servos attached to a catapult like device
 *  that throws a ball for a dog to fetch. This will also use serial communication
 *  so that two arduinos can communicate to each other. For this code, we will
control the
 *  servos--if the button on this arduino is pushed, it will reset the servos and
clear the lcd screen.
 *  If the button on the other arduino is pushed, we will read it in from this
program and move the servo to throw.
 *  Once the button is pushed to throw, "FETCH!" will appear on the lcd and a 1
will be
 *  written to the serial monitor for the other arduino to read and light up its
green LED
 *  and sound the buzzer.
 */

#include <LiquidCrystal.h>
#include <Servo.h>

// CREATE SERVO OBBJECTS
Servo servo;
Servo servo2;

// VARIABLES TO HOLD PINS FOR BUTTONS AND LED
const int buttonPin = 7;
const int ledPin = 13;

// INITIALIZE THE LCD WITH PINS
LiquidCrystal lcd (12, 11, 5,4,3,2);

// VARIABLES TO HOLD THE BUTTON STATES TO NOTIFY IF BUTTONS HAVE BEEN PRESSED
int curButtonState;
int prevButtonState;

// VARIABLES TO HOLD LED STATE
int ledState = 0;

// VARIABLES TO KEEP TRACK OF SERVO POSITION
int servoPos = 0;
```

```
void setup() {
  // SET UP SERIAL
  Serial.begin(9600);

  // INITIALIZE THE SIZE OF LCD TO START
  lcd.begin(16, 2);

  // ATTACH THE SERVOS BASED ON THE PIN NUMBERS
  servo.attach(10);
  servo2.attach(9);

  // SET THE BUTTON AS INPUT AND THE LED AS OUTPUT
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);

  // UPDATE THE BUTTON STATE TO READ IN IF IT HAS BEEN PUSHED OR NOT
  curButtonState = digitalRead(buttonPin);
}

void loop() {

  // UPDATE THE PREVIOUS BUTTON STATE
  prevButtonState = curButtonState;

  // READ IN THE NEW STATE OF THE BUTTON AND UPDATE CURBUTTONSTATE
  curButtonState = digitalRead(buttonPin);

  // CHECK IF THE SERIAL GOT ANY INPUT--IF BUTTON WAS PRESSED IT SHOULD RECEIVE A 1
  if(Serial.read() == 1) {

    // UPDATE THE LED STATE TO THE OPPOSITE OF WHAT IT WAS BEFORE
    ledState = !ledState;

    // REFLECT IT ONTO THE LED BY WRITING TO IT
    digitalWrite(ledPin, ledState);

    // SET THE CURSOR TO 0TH ROW AND 0TH COLUMN
    lcd.setCursor(0,0);

    // WRITE "FETCH" ONTO THE LCD
    lcd.print("FETCH!");

    // MOVE THE SERVOS
    servo.write(135);
    servo2.write(135);

  }
```

```
  // CHECK IF THE BUTTON HAS BEEN PRESSED
  if (curButtonState == HIGH && prevButtonState == LOW) {

    // WRITE IN A 1 TO THE SERIAL TO COMMUNICATE
    Serial.write(1);

    // CLEAR THE LCD
    lcd.clear();

    // RESET THE SERVOS
    servo.write(30);
    servo2.write(30);
  }
}
```

SemProj2.ino Code:

```
/* semproj2.ino */

//
// <JUDY PARK, JULIANNE PABONA, SHIRLEY LI>
// SEMESTER PROJECT
// U. Of Illinois, Chicago
// CS 362: Fall 2020
//

/* Description: Code that uses two servos attached to a catapult like device
 *  that throws a ball for a dog to fetch. This will also use serial communication
 *  so that two arduinos can communicate to each other. For this code, we will send
 *  information through Serial to the other Arduino once the button on the Arduino
is pushed to
 *  reset the servos and turn the red LED on. If the button connected to this
Arduino is pushed,
 *  it will make the buzzer go off. The volume of the buzzer could also be
controlled via the potentiometer.
 */

#include <LiquidCrystal.h>

/* VARIABLES TO RUN PROGRAM */
// Variable to hold the pin numbers for buttons and leds
const int buttonPin = 7;
const int ledPin = 13;
const int buzzerPin = 6;
const int potPin = A2;

// Variables to hold the button states to notify if buttons have been pressed
int curButtonState;
int prevButtonState;

//Variable to hold potentiometer value
int potVal;

// Variables to hold the led state
int ledState = 0;

/* SETUP THE INPUTS AND OUTPUTS*/
void setup() {
  // Set up the Serial
  Serial.begin(9600);

  // Set the button as input and led as output
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(potPin, INPUT);
```

```
  // Update the button state to read in if it has been pushed or not
  curButtonState = digitalRead(buttonPin);
}

void loop() {
  // Update the previous button state
  prevButtonState = curButtonState;

  // Read in the new state of the button and update curButtonState
  curButtonState = digitalRead(buttonPin);

  // Check if the serial got any input--if button was pressed it should receive a 1
  if(Serial.read() == 1) {
    // Update the led state to the opposite of what it was before
    ledState = !ledState;

    // Reflect it onto the led by writing to it
    digitalWrite(ledPin, ledState);

    //write 2 to the serial to indicate a reset of servos
    Serial.write(2);
  }

  // Check if the button has been pressed
  if (curButtonState == HIGH && prevButtonState == LOW) {

    // Write in a 1 to the serial to communicate
    Serial.write(1);

    //set potVal to the value at the potentiometer
    potVal = analogRead(potPin);

    // map the values stored in potval to store into a variable holding what the
volume buzzer should be (0 to 6)
    byte volume = map(potVal, 0, 1024, 0, 6);

    // change volume of buzzer
    analogWrite(buzzerPin, volume);
    delay(3000);

    //stop the buzzer from making sound
    analogWrite(buzzerPin, 0);
  }
}
```