NEW YORK CITY COLLEGE OF TECHNOLOGY
CITY TECH

Department of Computer Engineering Technology

Final: Arcade Claw Machine (prototype)

Course: EMT 2461 Electromechanical Systems
Section:
Semester: Fall 2024
Instructor: Mauricio Cardenas MS CEE.
Lab #: Final project
Student: Ronny Yeap, Yusuf Smaili

Date: 12/18/2024

Contents

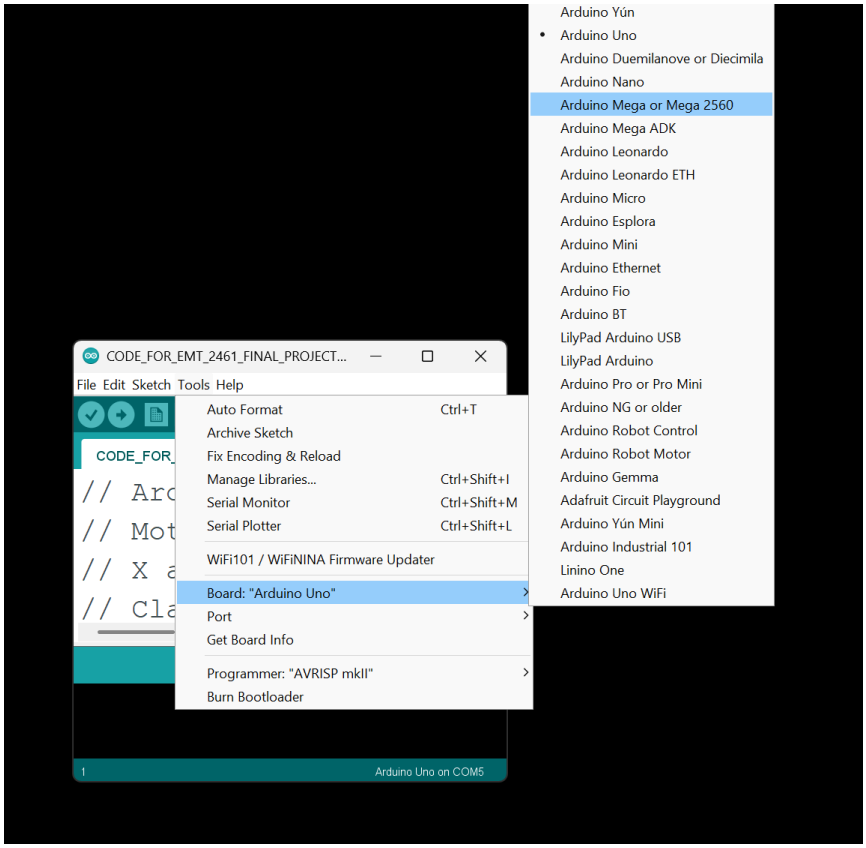
Objective

Our goal is to create an Arcade Claw Machine prototype using an Arduino MEGA with DC Motors. Since this is a prototype, *it does not have to be perfect.*

Background

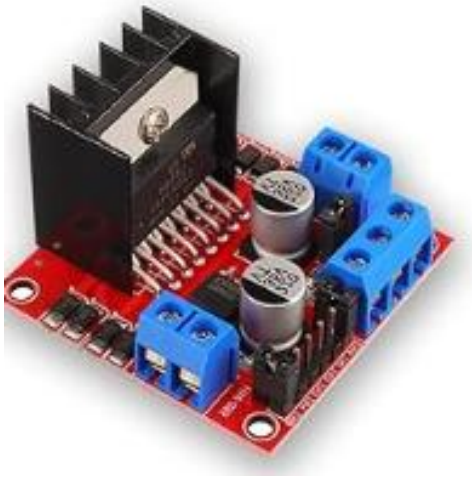
An arcade claw machine is a coin-operated game often found in arcades. It consists of a glass enclosure filled with prizes such as plush toys, gadgets, candy, or collectibles. Players use a joystick and buttons to control a mechanical claw, trying to grab and retrieve an item. In this lab, we simulate the claw machine using Arduino IDE. Arduino IDE is free, open-source software for writing and uploading code to an Arduino board. It supports multiple operating systems including Windows, Mac OS, Linux. So, you can duplicate this project with concern for free, just make sure the version of the Arduino is up to date so the library will work.

The Arduino board we are using is the Arduino Mega. The Arduino Mega serves as a central microcontroller unit for complex systems that require multiple I/O connections. It is based on an ATmega2560 microcontroller and operates at a 5V logic level. It provides 54 digital I/O pins, 6 analog inputs, and 4 UARTs for serial communication. The higher number of pins and memory makes it ideal for projects with multiple sensors, motors, and communication interfaces. It uses PWM signals for motor speed control and digital signals for direction. When you code the Arduino IDE using the Arduino Mega, make sure you change the board info in the Arduino IDE and choose the correct board, some Arduino IDE will detect, some will not. You can make sure the board info is correct by going to tools >> board >> “board name”. If you are using an Arduino Mega, the name will be “Arduino Mega or Mega2560”.



If you want to use Arduino Uno it is fine but be aware that only some pins have PWM modules. Arduino mega can be expanded by adding an LCD that needs more pins to show user what to do and a coin detector to accept the coin and adding change to play the machine etc.

Other than that, L298N DC Motor Drivers are used to control the speed and direction of the DC motor. The L298N is a dual H-Bridge driver IC capable of independently controlling two DC motors. It receives PWM signals for speed regulation and logic signals for direction. This helps to control the DC motor movements left, right, forward, backward, up, and down, depending on what axis that DC motor is at. It runs by an external power source (6-12V) and features protection diodes, capacitors to manage back EMF generated and spike consumption by the DC motors. The external power source can be a 9V battery, but we decided to use the external source from the wall outlet with a 9V adapter because it is more consistent, and you do not need to worry about battery life.



Moreover, DC Motors contains a gearbox. The DC motor is used to manage the motion for the X, Y, and Z axis of the claw machine as a gantry and control by a joystick module. DC motors function by converting electrical energy into rotational motion through electromagnetic induction. Their speed is controlled by PWM signals, and their direction is managed by switching polarity using the H-Bridge circuit. Although the DC motor has a gear box, we still need to use PWM to control the speed but the pros of using DC motor with gear box is the torque of the DC motor is higher than regular DC motor.

Next, what is used is a joystick module as an input to control the DC motor and servo claw that will grip the object in the machine. The joystick simulates two potentiometers for X and Y axis via the gantry, it produces a range of voltages that correspond to movement speed. These signals are read as analog inputs. The integrated push-button generates a digital signal to trigger an action when pressed (a series or action that we will explain in the code section).

Then, a servo motor is used, which controls the claw's gripping mechanism with precise angular movements. Servo motors utilize closed-loop control to maintain a desired angular position. They interpret PWM signals, where the pulse width determines the rotation angle (e.g., 1ms for 0°, 2ms for 180°) (The servo we used only move 90 degrees so the PWM will be 1ms to 1.5ms). Servos are powered by a 5V supply lower power consumption compared to the DC motor.

Next, a timing belt and pulley is used for a mechanism that connects the gantry platform to move the DC motor to a precise location. A pulley is a wheel on a shaft that enables the timing belt passing over to move and change direction by using DC motor with gear box. The timing belt is a rubber belt that synchronizes the actions of the DC motor so the v-slot plate will follow the direction of the DC motor.

Below is the V-slot plate that will feed inside the aluminum extrusion and stick with DC motor so the DC motor will move x axis and y axis by using timing belt and pulley mechanism. The plate is feeding to the extrusion, so the plate will only move one direction precisely.

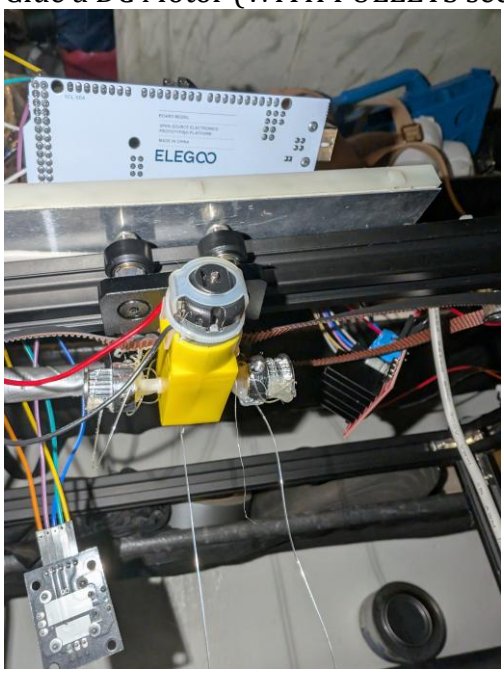
Below is the aluminum extrusion 2020. This extrusion is an excellent choice because it is cheaper and functions like other extrusions. We used 400 mm (about 1.31 ft) extrusion and 400 mm (about 1.31 ft) extrusion to create a frame and gantry for the machine.
T-mounting plates were also used to set a platform for the pulley and DC motor with gear box.

Procedure:

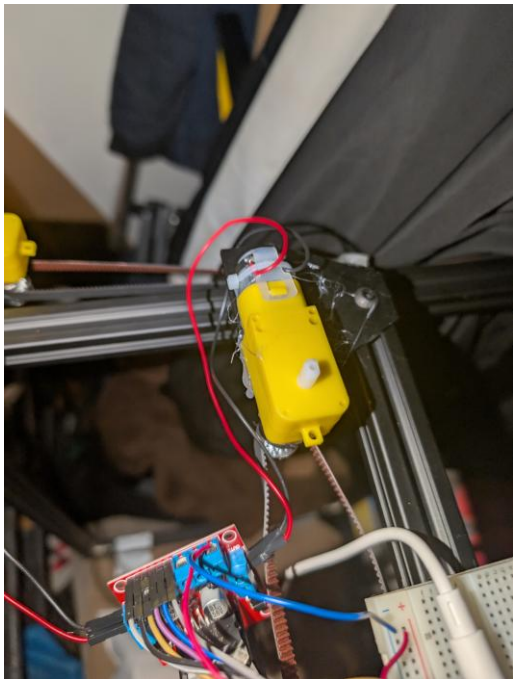
1. Use the Aluminum Extrusions to create a frame as shown below.

2. Tape or glue cardboard to the bottom of the frame and cut the cardboard to fit the frame.
3. Install V-Slot wheels (2) onto the two sides of the top of the frame

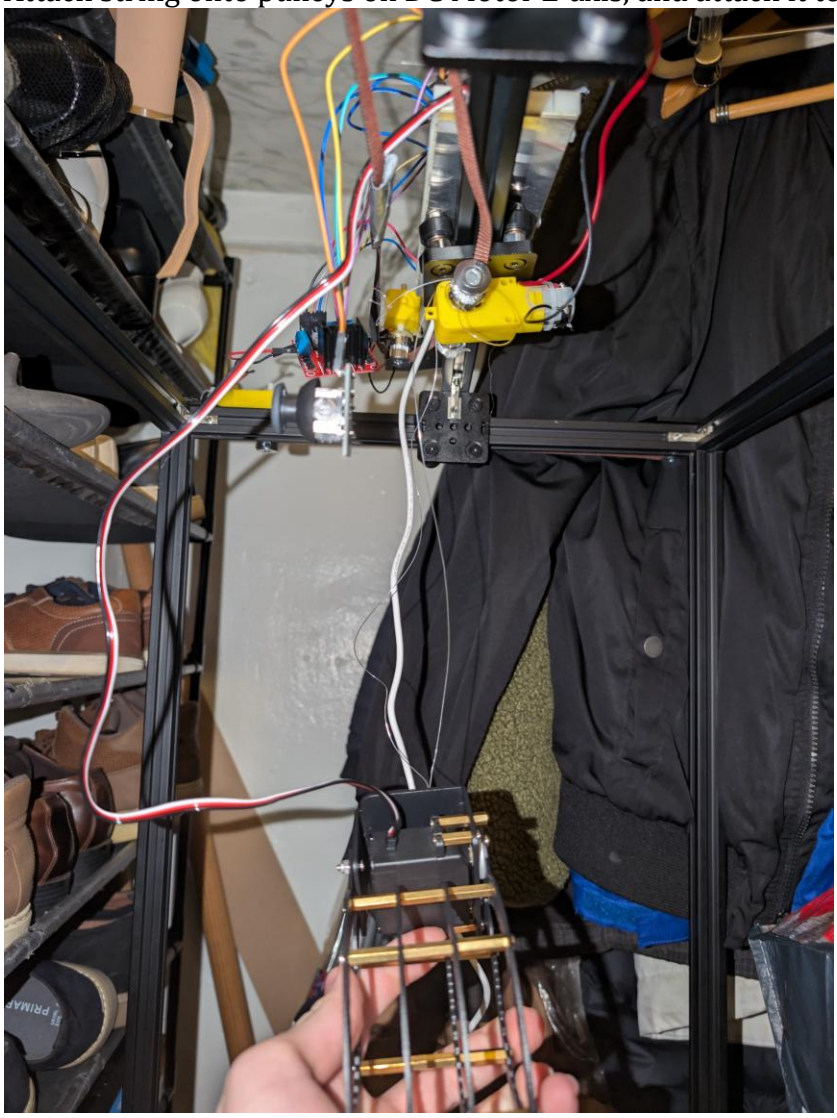
4. Take another aluminum extrusion and install another V-Slot wheel onto the frame (this will be used as the Z-Axis).
5. Glue a DC Motor (WITH PULLEYS see step 6) onto the Z-Axis V-Slot wheel.



6. Hot glue the pulleys onto the DC-Motor
7. Using the T-mounting plates, mount plates on the two ends of the top frame near the V-Slot wheels (this will be used to hot glue the DC Motors on top, as well as the pulleys.).



8. Attach string onto pulleys on DC Motor Z-axis, and attach it to the Servo

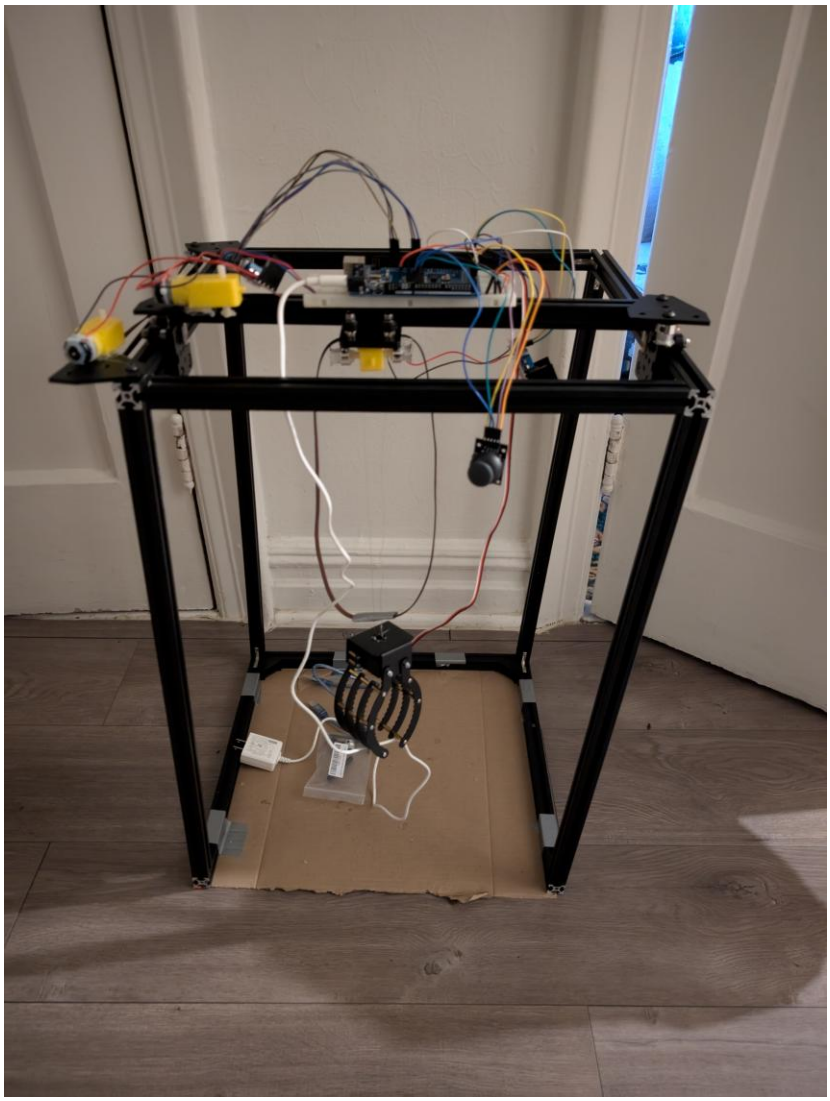


9. Be sure to attach belts onto the pulleys, keep in mind that the belts must be tight or else they will skip.
10. Wire your circuit accordingly (see electrical technicalities for schematics).
11. Implement code accordingly (see electrical technicalities for code).

Prototype End Result & Electrical Technicalities

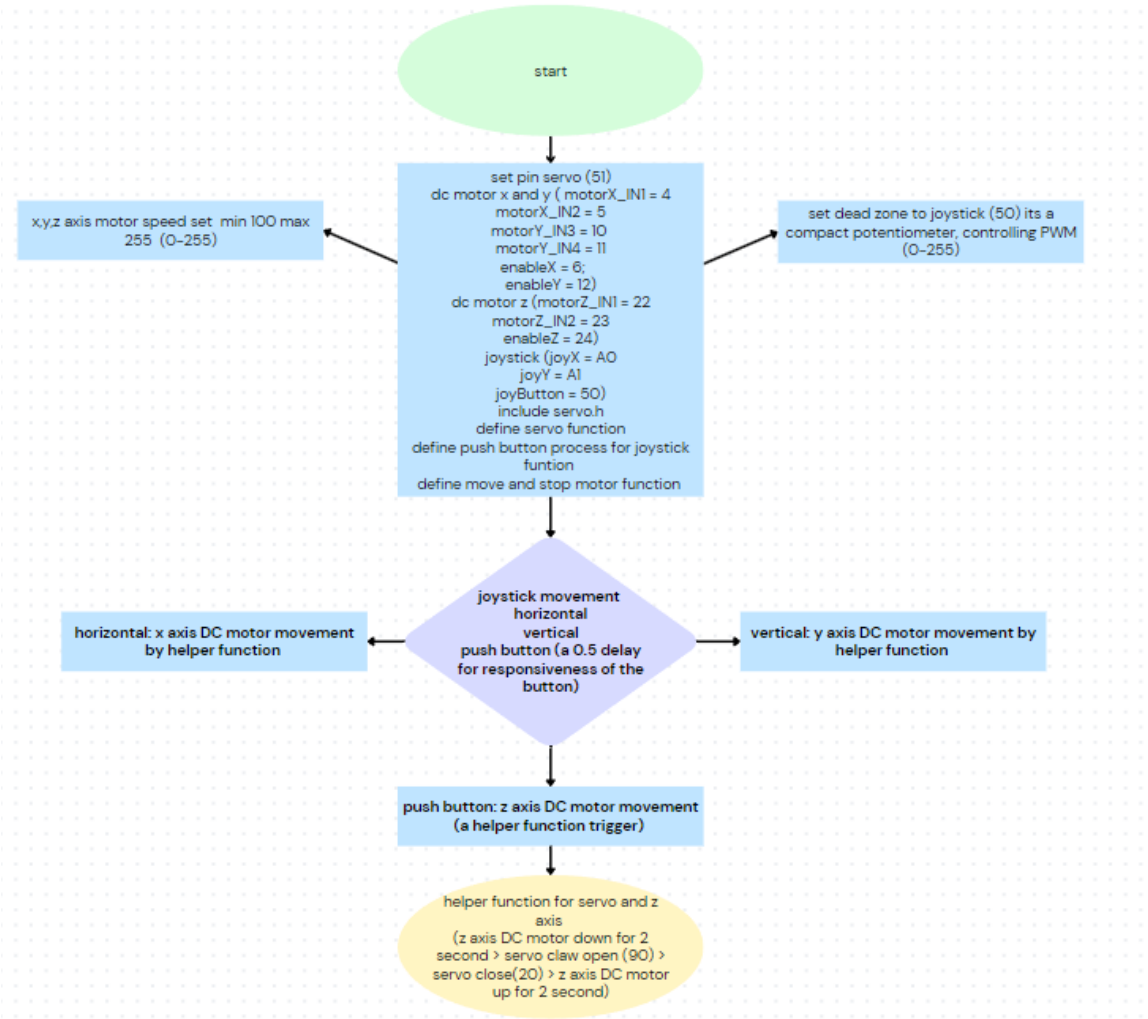
Explanation: This Arcade Claw Machine prototype will be using an Arduino MEGA since an Arduino MEGA provides more inputs (has more pins). The following information below will show all the technical details needed to understand the wiring, coding, and structure of the Arcade Claw Machine itself. Along with that, the result will also be shown to understand how the machine functions.

Prototype End Result:



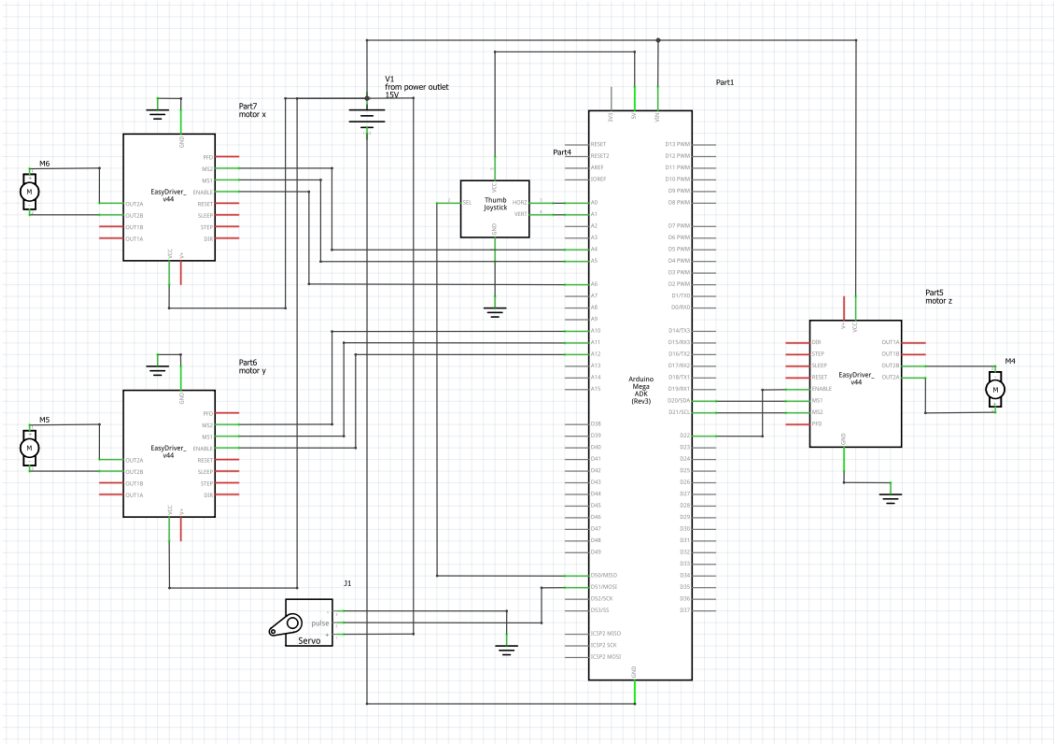
Explanation: On top of the frame, you can see the breadboard and Arduino with multiple connections. The joystick is also hanging in front of the prototype, with the power source being a wall plug. This wall plug connects to the Arduino, giving the DC Motors power through the "Vin" part of the Arduino board, which is then connected to the breadboard. The servo is powered by the 5V on the Arduino board and the joystick. The beauty of the Arduino MEGA is that there's multiple 5V inputs to use!

Block diagram:



Explanation: Control logic forms the backbone of any robotic system, and this one is no different. Using a joystick as the input apparatus, the user can direct the three DC motors in the system that control movement along the X-, Y-, and Z-axes. A push button functions as the instruction to the servo motor and Z-axis motor control system. The joystick is mounted to a plastic base, which itself is rigidly mounted to the Z-axis motor. The user can only move the joystick in and out from the plane parallel to YZ (the user can also imagine holding the base as the Y-axis and the joystick as the hand that, in a normal grasp, would rest along the X-axis). On the base, next to the push button, is mounted a circuit board on which are found important components that support the execution of servos and motors in a quiet manner.

Schematic View of Arcade Claw Machine:



Note: that the battery we use is a 9V outlet adapter. The model of the driver is not the same because we did not find the model in fritzing. All the common ground is connected to the GND of the Arduino mega via breadboard.

Code with comments:

```
#include <Servo.h>

// Motor Driver Pins (L298N) for X and Y motors (shared driver)
const int motorX_IN1 = 4;
const int motorX_IN2 = 5;
const int motorY_IN3 = 10;
const int motorY_IN4 = 11;
const int enableX = 6;
const int enableY = 12;

// Motor Driver Pins for Z-axis
const int motorZ_IN1 = 22;
const int motorZ_IN2 = 23;
const int enableZ = 24;

// Servo Pin for Claw
const int clawServoPin = 51;
Servo clawServo;

// Joystick Pins
const int joyX = A0;
const int joyY = A1;
const int joyButton = 50;

// Speed Limits
const int speedMin = 100;
const int speedMax = 255;

// Servo Angles
const int clawOpen = 90;
const int clawClose = 20;

// Joystick Deadzone
const int deadZone = 50;
```

Note: define all pins for all electronic device and library function

```
void setup()
{
  pinMode(motorX_IN1, OUTPUT);
  pinMode(motorX_IN2, OUTPUT);
  pinMode(motorY_IN3, OUTPUT);
  pinMode(motorY_IN4, OUTPUT);
  pinMode(enableX, OUTPUT);
  pinMode(enableY, OUTPUT);

  pinMode(motorZ_IN1, OUTPUT);
  pinMode(motorZ_IN2, OUTPUT);
  pinMode(enableZ, OUTPUT);
  pinMode(joyButton, INPUT_PULLUP);

  clawServo.attach(clawServoPin);
  clawServo.write(clawOpen);

  Serial.begin(9600);
}

void moveMotor(int enablePin, int IN1, int IN2, int speed, bool forward)
{
  analogWrite(enablePin, speed);
  digitalWrite(IN1, forward ? HIGH : LOW);
  digitalWrite(IN2, forward ? LOW : HIGH);
}

void stopMotor(int enablePin, int IN1, int IN2)
{
  analogWrite(enablePin, 0);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
}

void controlClaw(bool close)
{
  clawServo.write(close ? clawClose : clawOpen);
  delay(500);
}
```

Note: Setup all motors, define servo and DC motor as output and the pin it uses. Joystick as input and the pin uses. Initializes serial communication at 9600 baud for serial monitor purposes so we can debug the code. Define helper function:

moveMotor() function: Define parameter enablePin(motor's speed), IN1 and IN2 (motor direction) , speed (range from 0-255), forward (true = forward, false = backward)

stopMotor() function: enablePin(motor's speed) =0 and IN1 and IN2 (motor direction) = low

controlClaw() function: Boolean function to determine parameter of the claw close state is true or false.

```
void loop()
{
  int xValue = analogRead(joyX);
  int yValue = analogRead(joyY);
  bool buttonPressed = digitalRead(joyButton) == LOW;

  // Dynamically adjust speed
  int speedX = map(abs(xValue - 512), 0, 512, speedMin, speedMax);
  int speedY = map(abs(yValue - 512), 0, 512, speedMin, speedMax);

  // X-axis control with deadzone
  if (xValue > 512 + deadZone)
  {
    moveMotor(enableX, motorX_IN1, motorX_IN2, speedX, true);
  }
  else if (xValue < 512 - deadZone)
  {
    moveMotor(enableX, motorX_IN1, motorX_IN2, speedX, false);
  }
  else
  {
    stopMotor(enableX, motorX_IN1, motorX_IN2);
  }

  // Y-axis control with deadzone
  if (yValue > 512 + deadZone)
  {
    moveMotor(enableY, motorY_IN3, motorY_IN4, speedY, true);
  }
  else if (yValue < 512 - deadZone)
  {
    moveMotor(enableY, motorY_IN3, motorY_IN4, speedY, false);
  }
  else
  {
    stopMotor(enableY, motorY_IN3, motorY_IN4);
  }
  // Z-axis and claw control with button debounce
  static bool lastButtonState = HIGH;
  if (buttonPressed && lastButtonState == HIGH)
  {
    // Move Z Down
    moveMotor(enableZ, motorZ_IN1, motorZ_IN2, speedMax, false);
    delay(2000);
    stopMotor(enableZ, motorZ_IN1, motorZ_IN2);

    // Close Claw
    controlClaw(true);

    // Move Z Up
    moveMotor(enableZ, motorZ_IN1, motorZ_IN2, speedMax, true);
    delay(2000);
    stopMotor(enableZ, motorZ_IN1, motorZ_IN2);

    // Open Claw
    controlClaw(false);
  }
  lastButtonState = buttonPressed;

  delay(50); // Short delay for responsiveness
}
```

Note:

Joystick Inputs:

xValue reads the horizontal (X-axis) position of the joystick.

yValue reads the vertical (Y-axis) position of the joystick.

Button Input:
buttonPressed checks if the joystick button is pressed (LOW is press because of pullup configuration).
speedX and speedY parameter:
mapped to a range defined by speedMin and speedMax
yValue > 512 + deadZone : prevent unintentional movements when the joystick is idle.

Code Explanation: In this Arduino code, a robotic claw system with three direct current motors (DC) and a servo motor is under control, and its operation relies on a simple joystick and push button. When the code begins execution, it first performs a setup that initializes different elements: It sets the digital pins that communicate with the motor drivers, the servo, and the joystick. In addition, it defines several constants, including the speed at which the motors should run (a slow speed to allow for better control) and the angles at which the servo should rotate. The demo loop then gives basic control of the x and y-axis motors to the joystick, allowing them to move the claw in the horizontal and vertical planes. When the button is pressed, the system simulates groping: The z-axis motor moves the claw down, the clamp closes, the z-axis reverses, the clamp opens, and the sequence starts again.

Conclusion: Overall, this prototype took a lot of creativity and problem solving but came out with a promising result. There was a lot to learn, and tons of useful information for the future to come. This project within itself is a great starting point for the future of a computer engineer.

Testing Results and Analysis

Problems:

Overall, this project went smoothly on wiring and should not be hard but we are facing a short length of wire so the wire will be messy and tangle each other when we are controlling the project. We recommend that you should have a bunch of longer wires when you want to redo this project.

Coding is not easy when you are not using a function because it is going to block the entire code, so we advise you to use a helper function that will not block the code from running smoothly.

Note that the DC motor's speed can be controlled by using PWM from the range of 0-255. We use 150 because we feel that it is the proper speed. When you try to control the gantry, please feel free to adjust the speed that suits you.

A dead zone has been set for the joystick module because we found that when we do not touch the joystick, the gantry moves itself and causes the gantry to wiggle. After adding the dead zone, the dc motor will not move when we do not touch the joystick

and become more stable. The function triggered by the push button of the joystick can be adjusted depending on the height of your machine.

We also recommend that we prepare a bunch of jumper wires to duplicate this project because we were facing a problem that if the jumper wire were not that long enough to let our gantry move freely in the actual space we provide.

The belt and pulleys provided a major problem for us. If the belt is not tight enough, the pulley will not catch the belt, therefore not moving the gantry. We had to re-adjust the plates several times to tighten the belt, but also had to be careful of ripping the pulley as the hot glue has its limits.

Possible better solutions:

Managing wires: Use longer wires of better quality so they do not tangle. Also, make sure the wiring is clean, reducing the mess of wires around the machine and improving control. Optimizing the code: Make sure you use non-blocking helper functions. If you do not, the program will freeze at certain points during operation, and I do not believe you want that.

Adjusting the speed: Use different PWM values. The ones you are using now seem to make the motor run too fast for the smooth, precise operation I suspect you want from the gantry.

Implementation of the Dead Zone: Incorporate a joystick dead zone in the design phase to avoid unintended motor movements and to increase system stability. Preparation of Components: Secure ample amounts of jumper wire and other necessary components so that movement within the working space is not limited in any manner.

Belt and Pulley System: Use better methods of belt tightening, like adjustable tensioners, to avoid the constant need for re-tightening and the not-so-simple solution of using hot glue, which can fail under stress. These methods would both enhance and improve the user-friendliness and replicability of the project, allowing for a more reliable and functional result.

Conclusions

The Arcade Claw Machine prototype was a marvelous learning experience. The sheer difficulty of the undertaking forced us to think creatively and solve problems in ways we had not imagined. We wrestled with the issues of wire management, belt tensioning, and code optimization. Nevertheless, we succeeded in bringing the project to life, powering it with an Arduino MEGA and using DC motors and servo mechanisms to achieve a more arcade-like experience. The organization and precision required in this project taught me a lot about adaptability and the problem-solving nature of engineering at large, in a way that made those principles feel

concrete. If I am to refine anything, I think I should explore better ways to manage wire systems, come up with better means of tightening belts and pulleys, and generally use coding as a means of achieving smoother operation in any project.