**Claw Arm on Wheels**

Date: October 22, 2019

Course: ENGR 1212 Section 006

Instructor: Professor Reddivari

Group Members: Emmanuel Delgado, Jason Greaves-Taylor, Robert Mulkey

**Executive Summary**

Our finished project is a device that contains a claw arm which can roll freely until it senses an object, reaches it and finally grabs it. We used components of an Arduino kit [servo, DC Motor, 4 Servo motors, one LED, one IR Obstacle Sensor, one toggle switch button and a breadboard] to build our device. In order to perform the necessary movement, the arm and the wheels on our device must be assembled on a breadboard. An established communication between the hardware and the software installed on a computer was necessary to upload. We created a sequence of code written on C programing language using the Arduino text editor and uploaded the compiled code to our device via the Arduino Mega controller. Using Inventor software, we will make 3 additional part files that we 3D printed that served as arm, wheel and claw. The DC motor is mounted at the bottom of a 3D printed frame to turn a 3D printed gear.

# **Table of Contents**

|  |  |
| --- | --- |
| **Section** | **Page Numbers** |
| Executive Summary | 2 |
| Table of Contents | 3 |
| Design Problem and Objective | 4 |
| Detailed Design Documentation | 5-15 |
| Conclusions | 16 |
| References | 17 |

# **Design Problem and Objective**

The problem that we are trying to solve with our design is the lack of small-scale claw arms that can drive themselves to and object and pick it up. Many small-scale claw arms are made with a fixed base, some examples are the LewanSoul 6DOF Arm STEAM Robot Scratch Arduino and the OWI Robotic Arm Edge. Since these examples have a fixed base, they can only pick up objects which are within the maximum limit of the range of their arm’s reach, our design intends to fix this. By mounting our claw arm on wheels, we will overcome the flaw in previous designs by allowing our arm to roll until it reaches an object pick up. The final design that we’re going for will have 3 degrees of motion in the arm and roll forward under its own power. In the final design after the toggle switch is pressed the arm will roll forward until an IR sensor senses an object. The arm will then stop grab the object and attempt to pick it up.



**LewanSoul 6DOF Arm STEAM Robot Scratch Arduino**

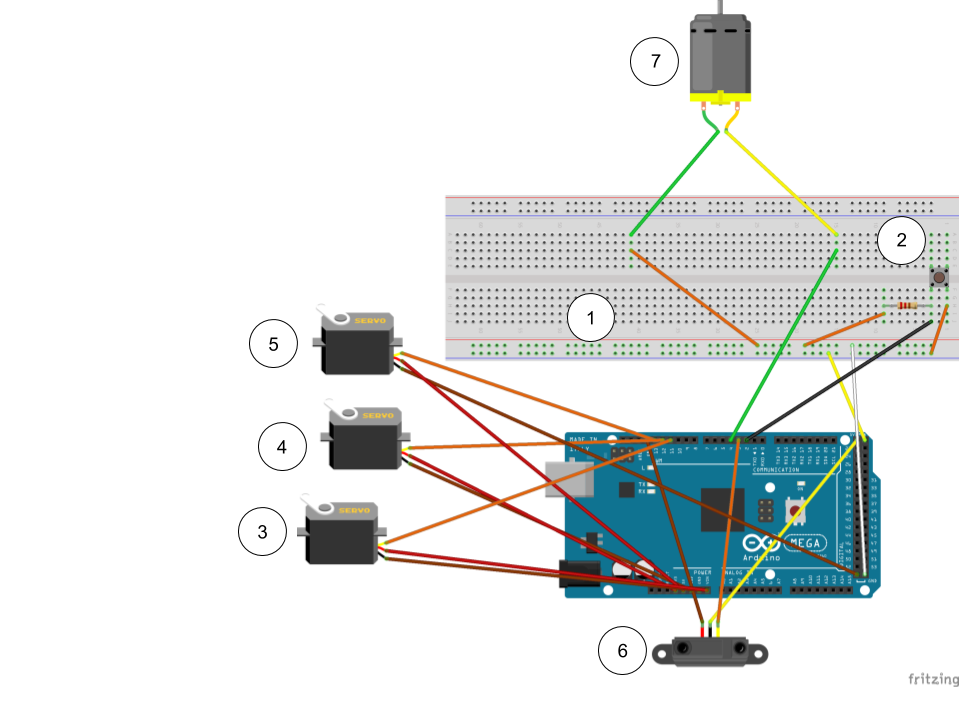


**OWI Robotic Arm Edge**

# **Detailed Design Documentation**

We have all the Arduino components necessary to power the claw arm. Additionally, the Arduino code can provide functions to allow each motor to rotate given a button input and the IR obstacle sensor’s detection of an object. A servo motor wired to the breadboard will rotate the entire arm about a vertical axis until an IR obstacle sensor that is wired to the board and attached close to the claw detects an object. Another servo motor will rotate the claw at higher or lower angles as the claw closes to pick up an object. The wiring from the diagram below should allow the motors to rotate in response to the press of a button and the sensor’s detection of an object.

**Breadboard Layout of Our Current Design**



1. One wire is connected to a 5V port on the Arduino and a negative terminal on the breadboard while another wire is connected to a ground port and a positive terminal on the breadboard. This ensures that a complete circuit is created on the breadboard to provide power for the components in our design.
2. A push button is wired to positive and negative terminals in the breadboard as well as a number port in order to delineate the button as an input component.
3. A servo motor has three ports that allow the motor to be wired to a voltage source port(5V), a ground port, and a number port for the motor to receive instructions from our code. This motor will act to rotate the entire arm horizontally between 0 degrees and 180 degrees.
4. This servo motor will rotate the frame holding the claw vertically as the claw tries to pick up an object.
5. This servo motor will rotate a gear in our claw which will allow the claw to close in order to pick up on object.
6. The IR obstacle sensor is wired in a similar fashion to the servo motor, which allows it to receive instructions from the code and be connected to a circuit. The sensor will detect an object, and this input will change the behavior of the claw arm as it begins to pick up an object.
7. The DC motor requires connection to a number port to act from the code, yet it can be connected to a positive terminal to complete the circuit. The DC motor will move an axle with wheels based on instructions the code provides.

We have also produced Arduino code shown below that should allow the servo motors to rotate depending on their functions in only one instance of a button press while an obstacle sensor detects for an object.

**Arduino Code for Our Current Design**

//The servo library is imported to allow the program to call servo functions.

#include <Servo.h>

//Variables for each servo motor are initialized.

//This variable allows for control of the servo motor that rotates the entire arm about a vertical axis.

Servo servo1;

//This variable allows for control of the servo motor that rotates the upper half of the arm about a horizontal axis.

Servo servo2;

//This variable allows for control of the servo motor that rotates gears which cause the claw to close or open.

Servo servo3;

//An angle value for the servo motors' angular position is initialized.

int position = 0;

//Variables are initialized based on each circuit component's port.

int button = 2;

int obstacleSensor = 3;

int dcMotor = 4;

//Other variables are initialized for use in the program.

int start = 0;

int lastPos = 0;

int moveTime = 1000;

int runTime = 3000;

int dropAngle = 30;

int i = 0;

void setup()

{

// put your setup code here, to run once:

//The button and obstacle sensor are set as input components while the motors are set as output components.

pinMode(button,INPUT);

pinMode(obstacleSensor,INPUT);

servo1.attach(11);

servo2.attach(12);

servo3.attach(13);

pinMode(dcMotor,OUTPUT);

}

void loop()

{

// put your main code here, to run repeatedly:

//The program will check if the button is pressed as well as if the obstacle sensor detects an object.

int press = digitalRead(button);

//Code in this loop executes when the button is pressed.

if(press == HIGH)

{

//The start variable is set to one so that the main code is not dependent on the button always being held.

start = 1;

}

//Code in this loop executes when start = 1 immediately after the button is pressed.

if(start == 1)

{

//Code in this loop executes as the sensor does not detect an object.

//The arm will rotate back and forth for two iterations before moving forward.

for(i = 0; i < 2; i += 1)

{

//The angle of servo1 changes from 0 degrees to 180 degrees.

for(position = 0; position < 180; position += 1)

{

servo1.write(position);

//The moveObject function will execute whenever the sensor detects an object along the arm's rotation.

moveObject();

//Once the moveObject function executes, the program will leave the for loop and stop until the button is pressed again.

if(start = 0)

{

break;

}

delay(10);

}

//The angle of servo1 changes from 180 degrees to 0 degrees.

for(position = 180; position > 0; position -= 1)

{

servo1.write(position);

moveObject();

//Once the moveObject function executes, the program will leave the for loop and stop until the button is pressed again.

if(start = 0)

{

break;

}

delay(10);

}

}

delay(900);

//The DC motor starts rotating its axle for as long as the moveTime in milliseconds lasts.

digitalWrite(dcMotor,HIGH);

delay(moveTime);

digitalWrite(dcMotor,LOW);

delay(1000);

//Variable i is set to 0 to repeat the process until the sensor detects an object.

i = 0;

}

//Code in the else loop will continue running until the obstacle sensor detects an object.

}

//A function called moveObject is defined to check if the sensor detects and object so that the arm can pick it up and move it elsewhere.

void moveObject()

{

int obstacle = digitalRead(obstacleSensor);

//Code in this loop executes when the sensor detects an object.

if(obstacle == LOW)

{

//The value of servo1's position is recorded as lastPos.

lastPos = position;

position = 90;

//The angle of servo2 changes from 90 degrees to 150 degrees.

for(position = 90; position < 150; position += 1)

{

servo2.write(position);

delay(10);

}

delay(900);

position = 0;

//The angle of servo3 changes from 0 degrees to 30 degrees.

for(position = 0; position < 30; position += 1)

{

servo3.write(position);

delay(10);

}

delay(900);

position = 150;

//The angle of servo2 changes from 150 degrees to 90 degrees.

for(position = 150; position > 90; position -= 1)

{

servo2.write(position);

delay(10);

}

delay(900);

//The position value is overwritten with lastPos to start the rotation of servo1's gears from its last position.

position = lastPos;

//The angular position of servo1 increases if its last position is less than 90 degrees.

while(position < 90)

{

servo1.write(position);

delay(10);

position += 1;

}

//The angular position of servo1 decreases if its last position is greater than 90 degrees.

while(position > 90)

{

servo1.write(position);

delay(10);

position -= 1;

}

delay(900);

//The DC motor starts rotating its axle for as long as the runTime in milliseconds lasts.

digitalWrite(dcMotor,HIGH);

delay(runTime);

digitalWrite(dcMotor,LOW);

//The angular position of servo1 increases if its position is less than the desired angle to drop the object.

while(position < dropAngle)

{

servo1.write(position);

delay(10);

position += 1;

}

//The angular position of servo1 decreases if its position is greater than the desired angle to drop the object.

while(position > dropAngle)

{

servo1.write(position);

delay(10);

position -= 1;

}

delay(900);

position = 90;

//The angle of servo2 changes from 90 degrees to 150 degrees.

for(position = 90; position < 150; position += 1)

{

servo2.write(position);

delay(10);

}

delay(900);

position = 30;

//The angle of servo3 changes from 30 degrees to 0 degrees.

for(position = 30; position > 0; position -= 1)

{

servo3.write(position);

delay(10);

}

delay(900);

position = 150;

//The angle of servo2 changes from 150 degrees to 90 degrees.

for(position = 150; position > 90; position -= 1)

{

servo2.write(position);

delay(10);

}

delay(900);

servo1.write(position);

start = 0;

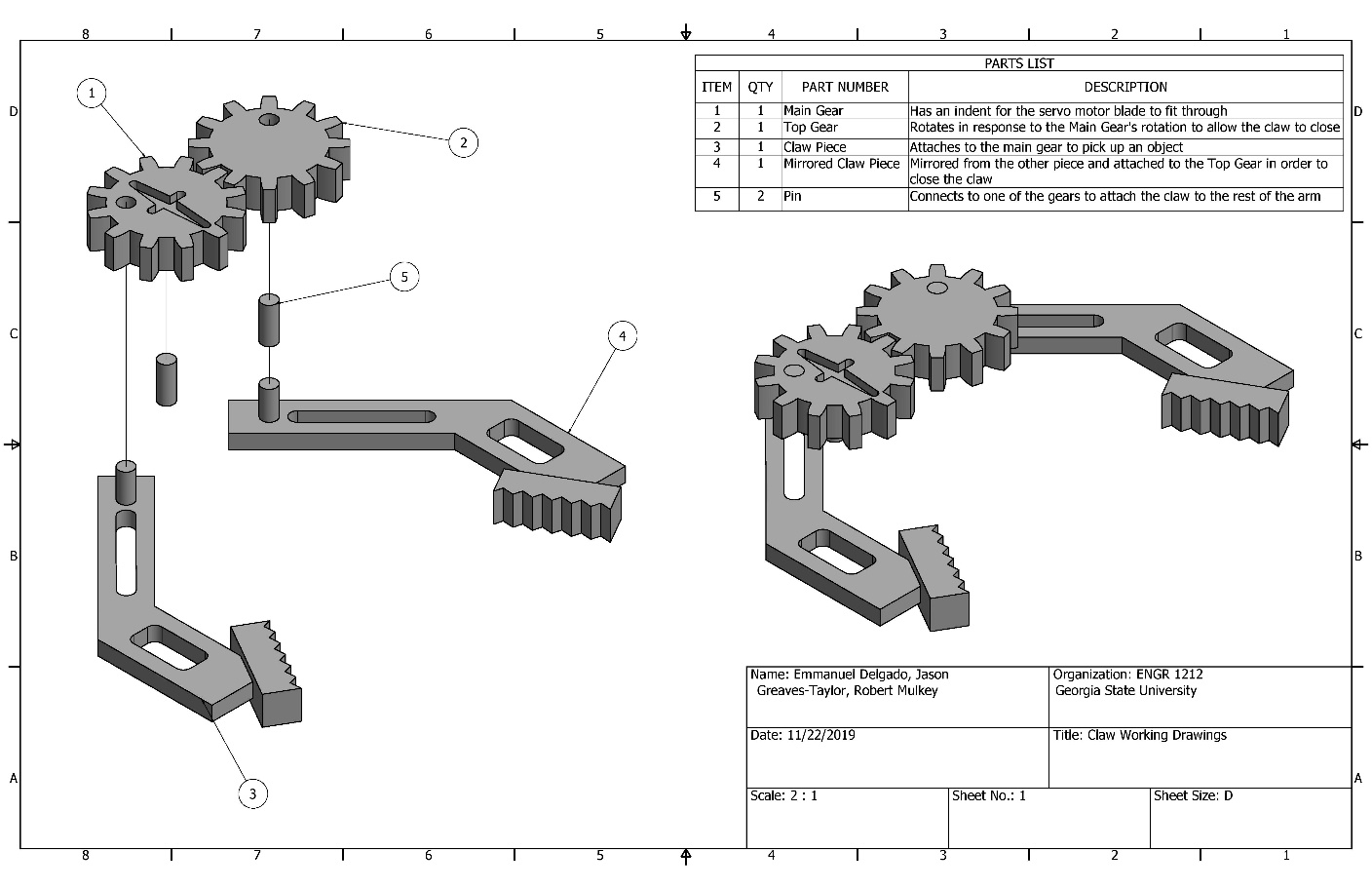
//The program will stop until the button is pressed again.

}

}

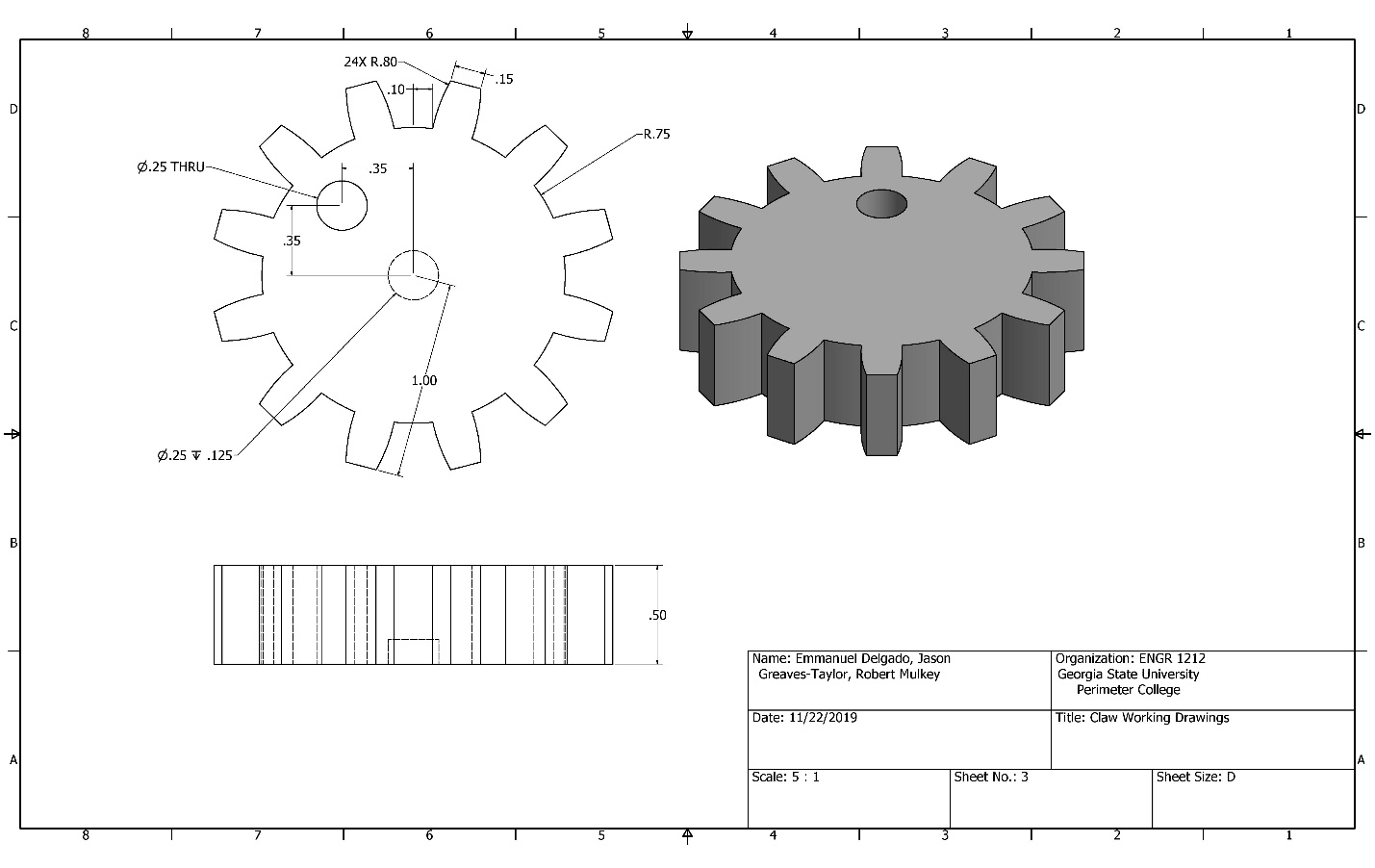
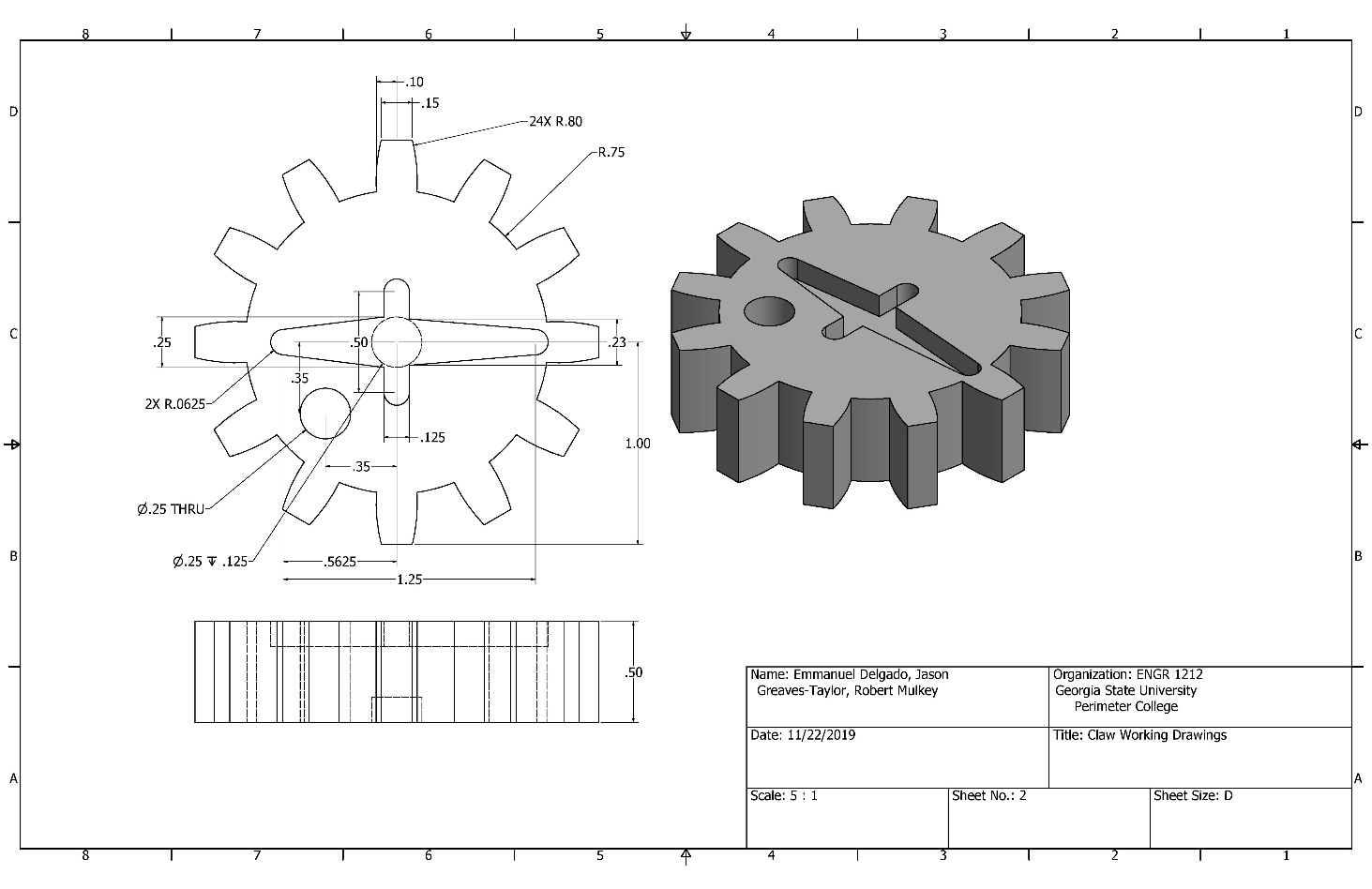
We have also designed parts of the claw arm on Autodesk Inventor for us to create a functional design in conjunction with our breadboard layout and code. We currently have a part that will act as our claw in our design, which will be attached to a servo motor to allow the claw to open and close with our current code. The function of this claw is to pick up and drop objects by rotating the gears which will move the claw tips closer to each other.

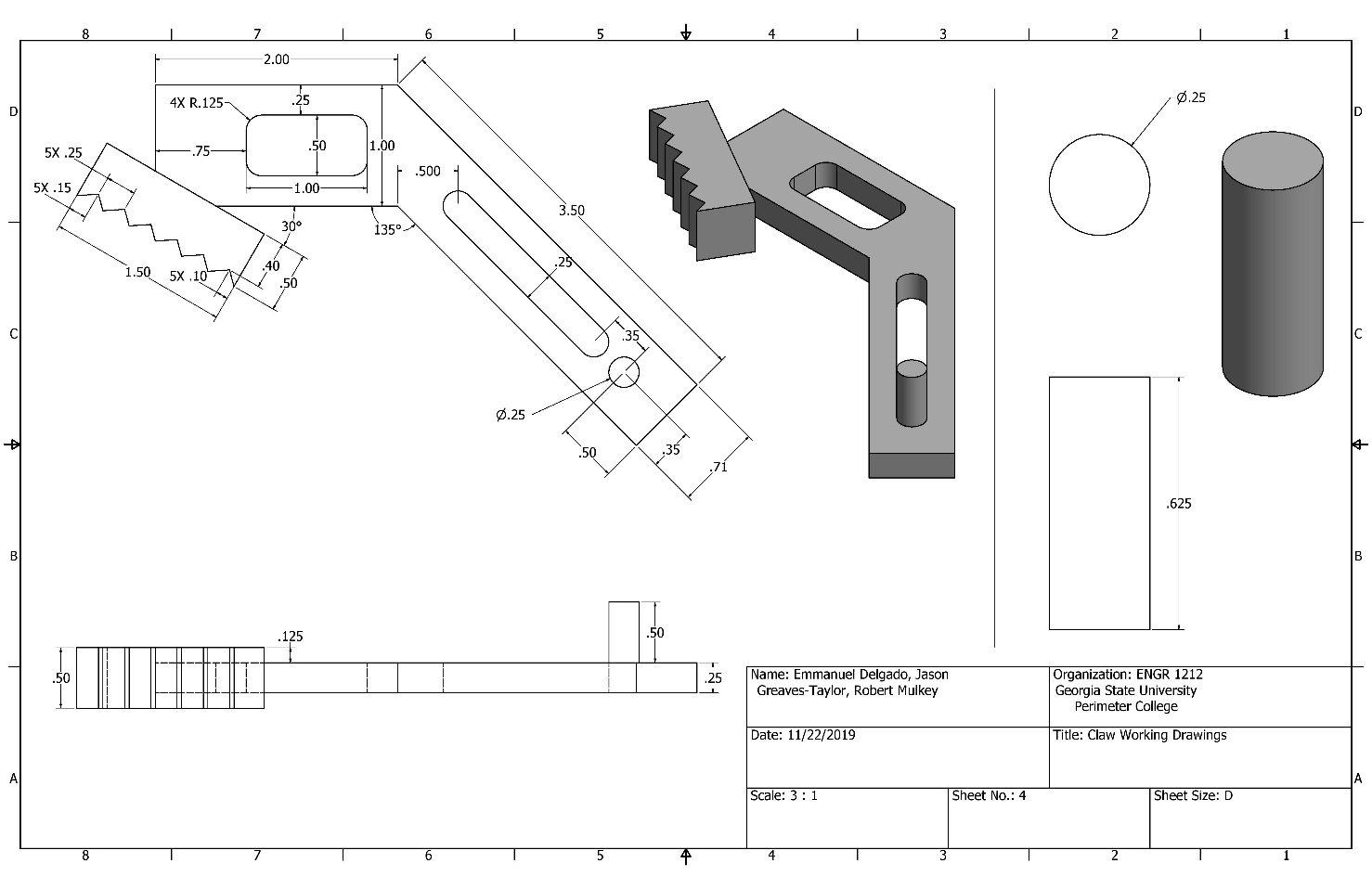
**Current Claw Assembly of Our Design**



1. A main gear is created with an indent in order to fit the servo motor’s blades into the gear. This will allow the gear to rotate along with the servo motor.
2. A top gear is placed with its teeth contacting the teeth of the main gear in such a way that when the main rotates in one direction, the top gear will rotate in the opposite direction.
3. This claw part extends past the gears in order to allow some distance between the object and the arm. The claw part is attached behind the center of the main gear as it will move closer to the other claw part when the gears rotate. The claw part can rotate alongside the gears as the claw picks up or drops the object. The tips of the claw are designed to provide a better grip on the object as the claw arm carries it. The tips are angled relative to the rest of the claw so that as the claw closes, the angle between the tips gets closer to zero.
4. This claw part is mirrored with respect to the previous claw part in order to fit into the top gear and allow these claw parts to move closer together to pick up an object.
5. Detachable pins are placed in the center of each gear to allow the claw to attach to the rest of our arm design.

The following working drawings of our claw assembly show the dimensions used in the part.

**Claw Part Drawings**



# A screenshot of a cell phone Description automatically generated

A close up of a piece of paper

Description automatically generated

A close up of text on a white background

Description automatically generated

# **Conclusions:**

With the given setup of the Arduino components, we can provide instructions to those components in order to allow our claw arm to function with the components. From our Arduino code, we can successfully program the behaviors of the motors in response to the obstacle sensor’s input. However, we are unable to reach the goals of our design specifications since we currently do not have the 3D printed parts necessary to assemble the claw arm. While we have designed components for our claw arm in Autodesk Inventor, we cannot determine whether these parts will function with our Arduino components until we have access to those 3D printed parts. Fortunately, once we have all the parts we need, we can attach each of them to their respective motors to completely assemble our claw arm. We can also adjust the layout of the servo motors and the obstacle sensor in our design to allow those components to properly function with our objectives. Additionally, we can slightly change the Arduino code depending on whether the servo motors can move their respective parts. Overall, any progress we make by adding the 3D printed parts will allow our design objectives for the claw arm to be met.

Beyond the progress we can make to accomplish our objectives with our claw arm design, we can improve various elements of the claw arm to increase the amount of functions it can perform. In addition to assembling all the components together into a claw arm, it is possible to add more degrees of freedom to our design to provide more movement options for our claw arm. For example, we can devise a system to allow the arm to rotate 360 degrees instead of the servo motor’s range of 0 to 180 degrees. Additionally, another DC motor can be configured into the design to allow the claw arm to move backwards in addition to its forward motion. Another potential extension to our design can utilize additional sensors to determine the size of the object in order to determine how much the claw should close. However, our focus on adding the necessary parts to complete this project under the given objectives means that such ideas will not be implemented until further development is needed.

# **References:**

*Amazon.Com: LewanSoul 6DOF Robotic Arm Kit for Arduino STEAM Robot Arm Kit with Handle PC Software and APP Control with Tutorial: Toys & Games*. [https://www.amazon.com/LewanSoul-Robotic-Arduino-Software-Tutorial/dp/B074T6DPKX. Accessed 25 Oct. 2019](https://www.amazon.com/LewanSoul-Robotic-Arduino-Software-Tutorial/dp/B074T6DPKX.%20Accessed%2025%20Oct.%202019).

*Amazon.Com: OWI Robotic Arm Edge | No Soldering Required | Extensive Range of Motion on All Pivot Points: Toys & Games*. <https://www.amazon.com/OWI-Robotic-Soldering-Required-Extensive/dp/B0017OFRCY/ref=pd_sbs_328_t_1/142-1473202-7845646?_encoding=UTF8&pd_rd_i=B0017OFRCY&pd_rd_r=611dd887-7b8f-4e04-b012-507fef0866c5&pd_rd_w=hyMMZ&pd_rd_wg=zelEO&pf_rd_p=5cfcfe89-300f-47d2-b1ad-a4e27203a02a&pf_rd_r=M0JHWX2TKN7EWSM3ZJ48&psc=1&refRID=M0JHWX2TKN7EWSM3ZJ48>. Accessed 25 Oct. 2019.