Peter Atsaves

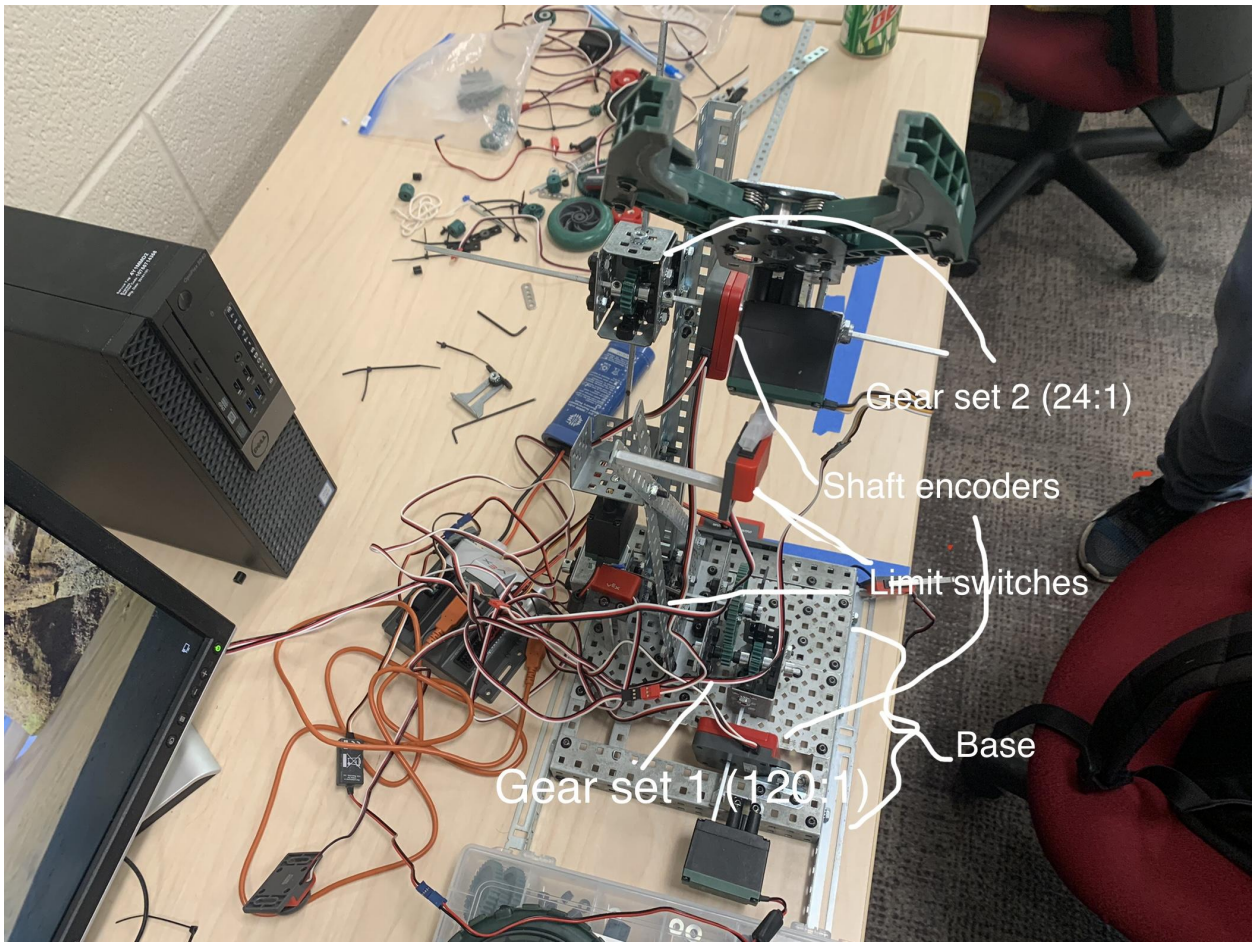
Introduction

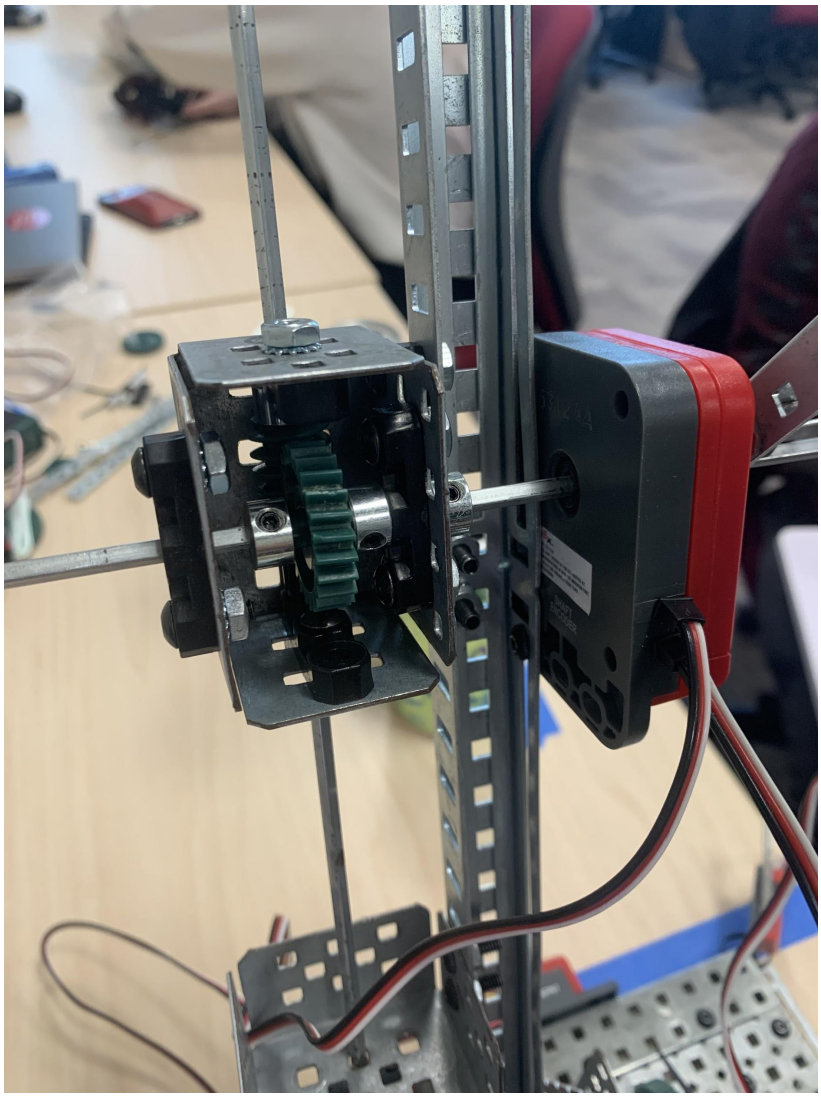
Our task was to create a robot that can pick up 3 cans, carry them up, and drop them. The robot would do this process for 1 can at a time and in one continuous run. We had to create the robot from scratch and make a 2 joint design that had a shoulder, bicep, forearm, and a claw at the end. Our approach to this project was to create a robot that has a big range of motion so it doesn't have trouble reaching any of the cans. We wanted to make a robot that was sturdy so we used a long arm for the bicep and had a base that is hard to move. We wanted to focus on physically building the majority of the robot first before we started implementing our software design. We felt that once we properly placed all the sensors and had a strong robot, coding the solution would be much easier.

Mechanical Design

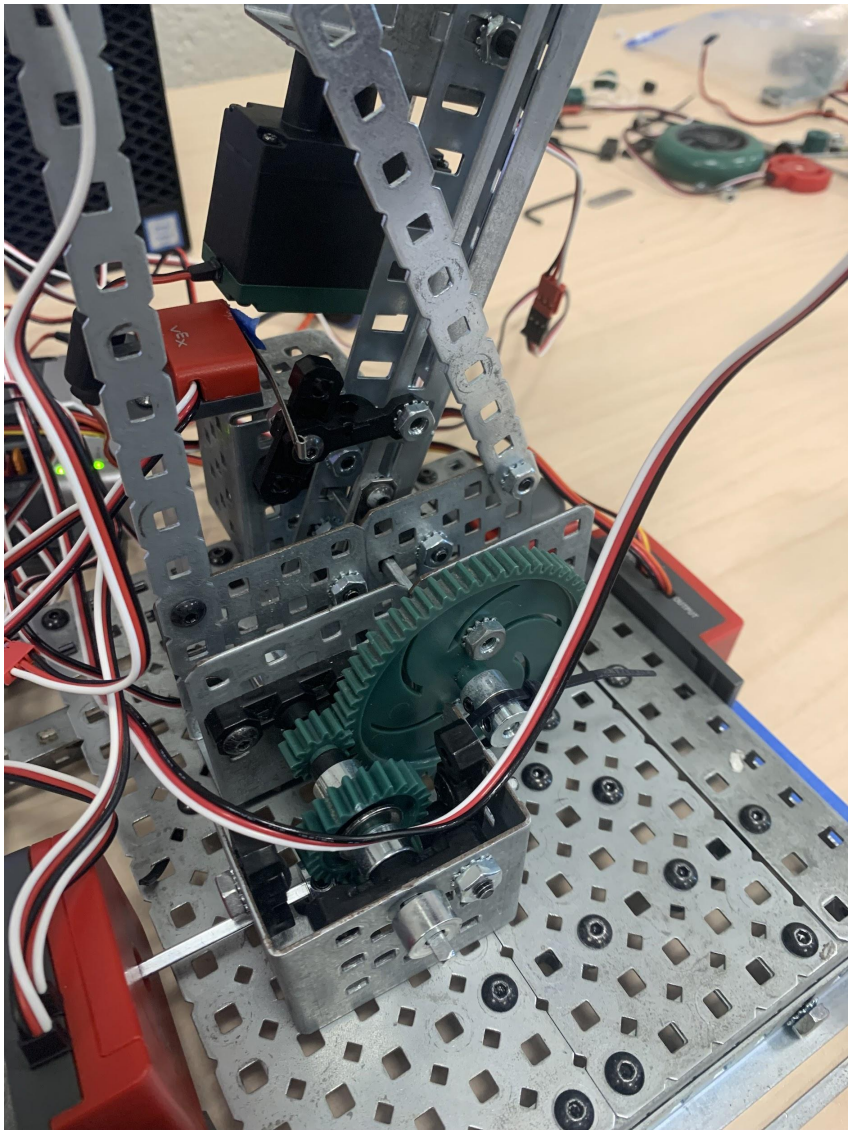
Our initial design started out very basic. We build a base made out of two metal sheets that were placed on two metal rails. Next we added the first set of gears. We built a worm gear by following the instructions, and then attached this to the base. We used a very long rod so that we could fit the shaft encoder on it and have enough room for the motor to be attached to the base and hold the rod in place. To build gear set 1, we used a 60-teeth gear and a 12-teeth gear and placed them so they are touching each other and both can rotate when one is moved. We had a rod that goes from the worm gear to the 12-teeth gear. So the motor is attached to the worm gear, this attaches to the 12-teeth gear which is attached to the 60-teeth gear and this gear has a rod holding the bicep which moves the bicep. So in essence, the motor is moving the bicep through all of these parts. We build this bicep by combining two long metal pieces. This ensured stability by putting them together. The rod that goes from the gear goes through a whole in the

long metal piece and is held in place by a cube that is screwed into the base. After working on the shoulder area, we move to the bicep. We attached a worm gear a couple inches from the end of the bicep. This worm gear is fastened onto the arm with screws. At the bottom of the bicep is a motor which is screwed to another metal cube. This cube is also screwed onto the bicep. Out of this motor is a rod that is attached to the worm gear which rotates the gears. When the bicep is standing up, the rod that is horizontal has a shaft encoder through it, on the other side of the bicep. Right next to this is the forearm. The forearm is one metal piece that is about half the length of the bicep. The forearm is held by the rod going through it. At the top of the forearm, a claw is attached to it with screws. The claw and forearm are facing the same way. The claw also has a motor to move its gears to open and close.


Gear set 2 (24:1)
Shaft encoders
Limit switches
Base
Gear set 1 (120:1)



This picture shows how the worm gear is placed on the bicep. This cube is screwed in, and the rod/axel extends all the way to the right for the forearm to attach. The shaft encoder is to the right on the other side of the bicep.



In this image, we can see how the gears are for the first set. A worm gear and a 12-teeth gear share a rod/axel. The worm gear is moved by a motor. The 12-teeth gear drives the 60-teeth gear.

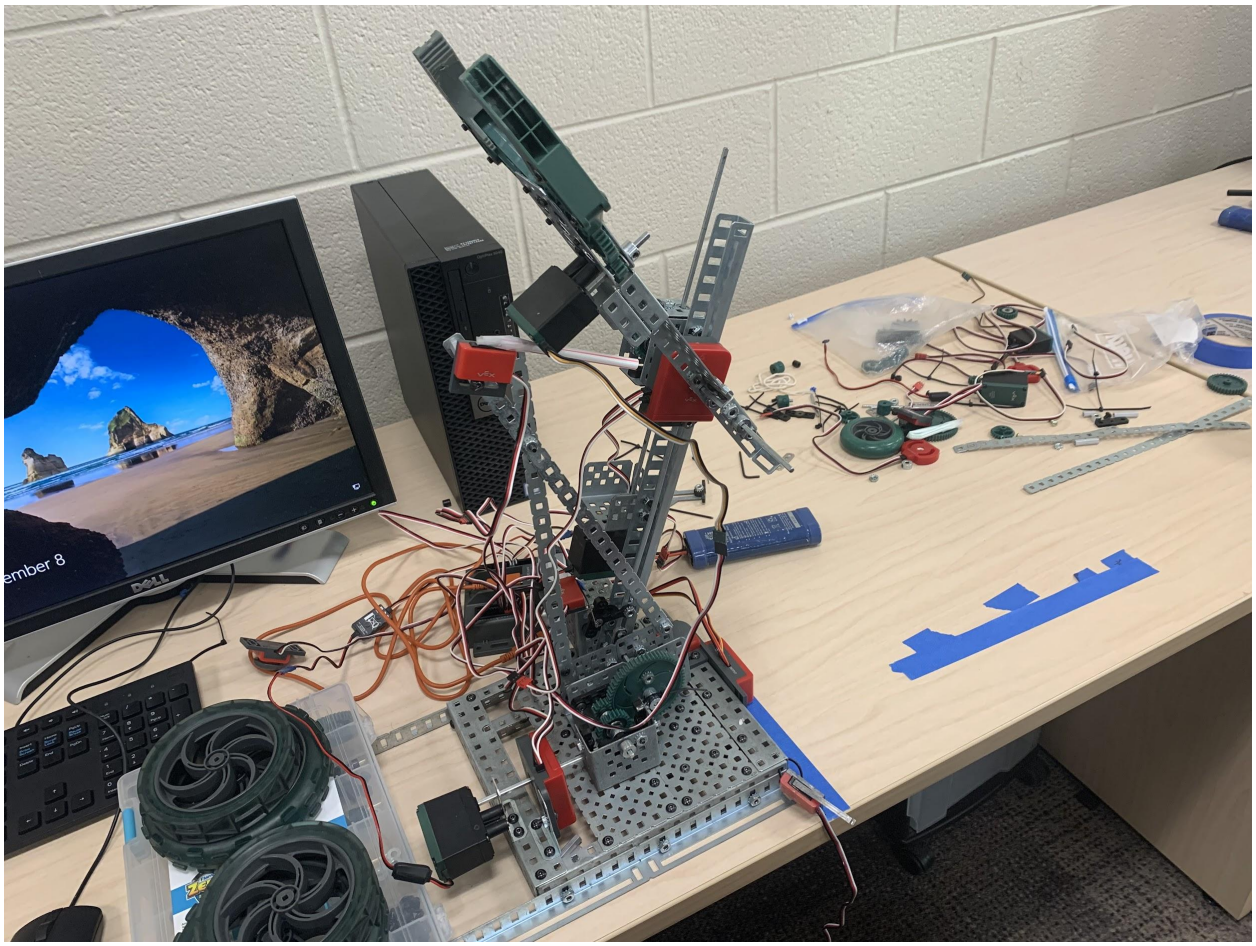
One modification is that we didn't realize that the worm gear was 24:1. We thought it was just 1:1, so initially we had 2 sets of 12 and 60 teeth gears on the base. This was a 600:1 ratio which was too much, so we got rid of one 12 and one 60 teeth gear. We also modified where we put our limit switches. We started with the limit switch on the base, but we realized it wasn't necessary for our bicep to go that far back. We moved it a couple times once we found how far our bicep

should go back. For the limit switch for the forearm, this took some thinking. We had to move this around too since we changed how far back the bicep would go. We used two thin metal pieces to stand up and hold the limit switch at an angle that would only be triggered when the can should be released. One modification that was annoying was that we had a lot of "give" in our arms. We didn't realize that this was because our worm gears weren't sturdy. We only used the cubes that had multiple openings, but we needed the cubes that are closed on all sides but one so that the rod/axel has somewhere to be held on multiple sides. By the time we realized this, we already had almost completed our robot, so we basically had to redo a lot of it and take a lot of it apart to do this. We also had an ultrasonic sensor in the front to measure the distance of the can. To determine the gears, we started with the first set. The worm gear is 24:1. This is attached to a 12-teeth gear and that is attached to the 60-teeth gear. This ratio is 5:1, so 24x5 is 120, making the ratio 24:1. The second set of gears is only a worm gear, since we didn't need as much torque because the forearm is lighter. This gear ratio is only 24:1 since that is what the worm gear is.

SoftwareDesign

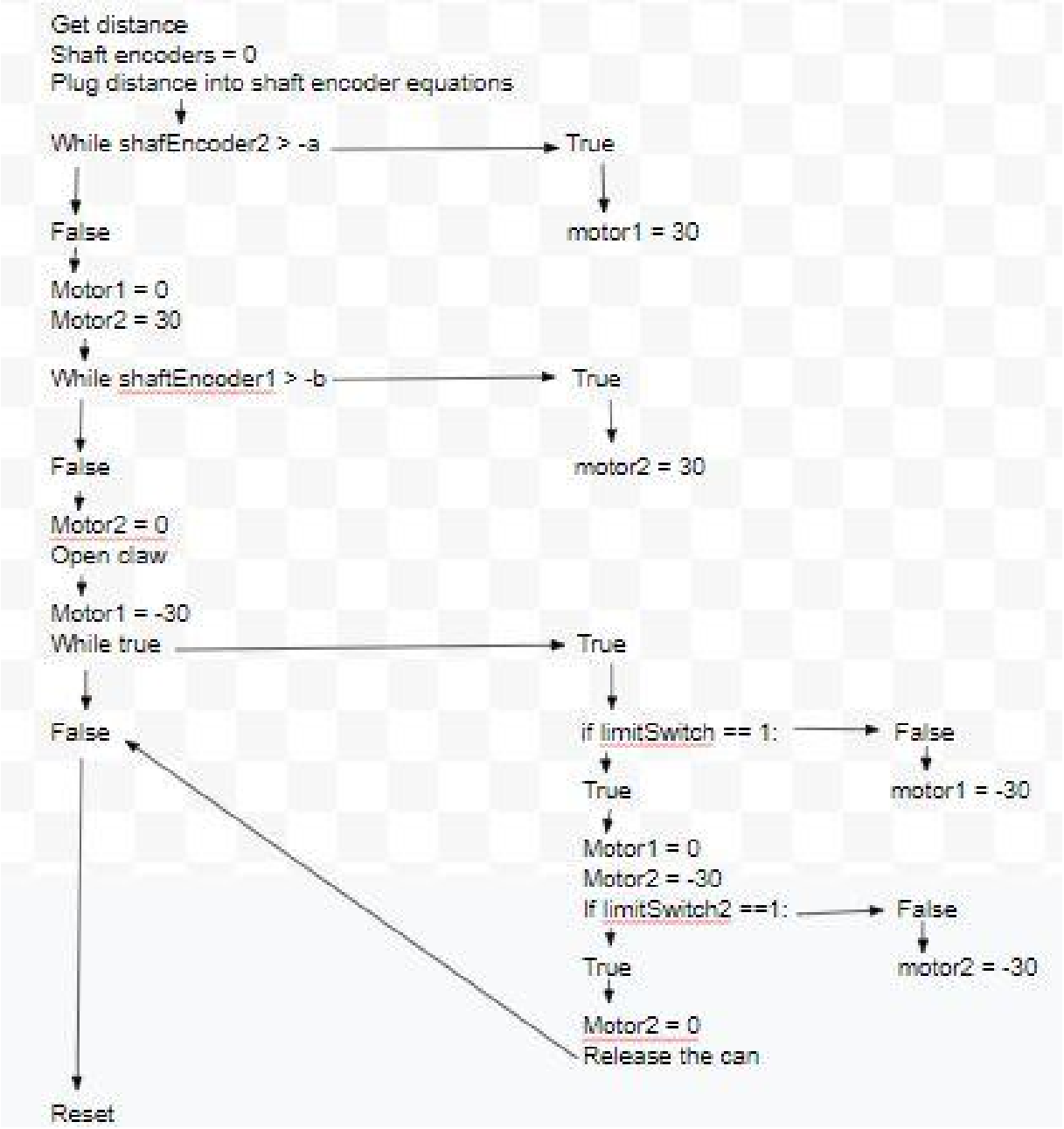
We spent the majority of our time working on the mechanical design of the robot, so it wasn't until the final few days that we significantly worked on the software design. Before that point, we still had some software. We had a limit switch and a bump switch so that if the bicep hit either one of them, it would stop moving. Once we added the forearm, we added the same thing in our code. Fast forward to the end of the project and this is when we added a lot more

software. At this point, we set a starting point for our robot.



This picture shows our starting point, where both limit switches are activated. We started by getting the ultrasonic sensor which is the distance of the can. We set the shaft encoders to 0. With the ultrasonic sensor, we plugged this into 2 equations which gives us the distance that our arms needed to move, called these values a and b. We moved the bicep until the shaft encoder was bigger than a, then the bicep stopped. Then the forearm moves until the shaft encoder is bigger than b. The claw that started out as open then closes to grip the can. The bicep starts moving the opposite way until it hits the limit switch. Once this happens, it stops, and the forearm moves until it hits its designated limit switch. The robot pauses for a second as the claw opens and

releases the can. The logic is also explained below. We didn't have to make many revisions because the logic was pretty simple.



Instead of using kinematics, we used a data-driven approach. We placed a can at a location and manually moved our bicep and forearm until we reached the can. We did this by constantly stopping the robot and changing the speeds of the arms on the software until we reached the designated location. To be consistent, we would move the bicep first and then lower the forearm to keep consistency since there are multiple ways you can get to the can using the arms. We also aimed for the middle of the can so that we have a good grip but also to keep consistent. Once we were at the location of the can, the claw would grab the can. The bicep would move back until it hit a limit switch and stop, then the forearm would move back until it hit a limit switch and stop. We gathered the shaft encoder value of each of the arms. We repeated this process for 5 different cans, making sure to get the start, end, and 3 places throughout. We also gathered the ultrasonic sensor value for the cans so that we had data for the distance of the can as well as how we need to move the arms. We used linear regression to generate an equation for the relationship between the ultrasonic sensor and shaft encoder 1, and the relationship between the ultrasonic sensor and shaft encoder 2. The equation we got for shaft encoder 2 was shaftEncoder2 = 24.72 x (distance) - 3574.6. The equation we got for shaft encoder 1 was shaftEncoder1 = 0.4074 x (distance) - 229.03. We then used these in our code so that we get the ultrasonic sensor value (distance of the can), then plug this value into both of our equations so the robot knows how far to move each of the arms.

Performance Evaluation

We didn't use kinematics but instead used the data-driven approach. We took 5 data points. The data is below, and we used this data to come up with 2 equations using linear

regression to predict the shaft encoder of the respective arm and how far it should move to pick up the can. ShaftEncoder2 = 24.72 x (distance) - 3574.6. ShaftEncoder1 = 0.4074 x (distance) - 229.03.

Ultrasonic Sensor	Motor 1 Shaft Encoder (SE2)	Motor 2 Shaft Encoder (SE1)
160	552	-161
202	1535	-144
282	2853	-122
312	3927	-110
360	5793	-72

This didn't lead to an entirely accurate equation since we only had 5 data points and there were still some times when the claw wouldn't grab the can in a good spot and drop it on the way up. Despite this, most of the time the equations were spot on in determining where the arms should move to pick up the can. Since we didn't do the kinematics, we don't know for sure the heights the robot can fix, but we know the distance. The ultrasonic sensor value must be between 160mm and 360mm. 160mm is when the bicep doesn't move and only the forearm does, and this is when the can is really close. 360mm is when the bicep and forearm have to extend out as far as possible. Any further and the claw can't reach the can. The robot succeeds really well in picking up cans at different locations. If I had to place a number, it would be close to 100% when we place the cans in line with the claw. Of these though, probably 10% of the cans are not picked up well and the claw is only holding the top of the can and these cans end up falling which isn't good. Another problem is that sometimes the claw goes too far down and is pushing the can down into the ground before lifting it. We can fix these if we had a claw that either rotates, or if we changed the direction of the claw. The claw right now is parallel with the forearm, but we

could make it a 90 degree angle instead. This way when we grab the cans, it is much more likely to get a good grip on the can since the claw will be coming horizontally from the side and not from the top. This allows consistent grabbing and gripping of the cans. This would also help if the shaft encoder equations aren't completely accurate. With the claw going horizontal instead of vertical, we can be slightly off in the distance, but the can will still be grabbed. The can might be moved out of location slightly, but the claw will still be able to get a grip on it. If we had more time and if our current solution failed, we would've implemented these changes. We wouldn't need any additional parts to make these changes either.

Conclusion

Overall, I am happy with the performance of the robot. We were able to get our robot to pick up all three cans and move the cans back and drop them. One thing we could've done to improve was to take more data. We only took 5 pieces of data, where we got the ultrasonic sensor value of a can, and the value of the 2 shaft encoders to reach this can. We used this to get an equation through linear regression, but we should have used more points. Sometimes when our robot picked up a can, it would pick it up perfectly, but other times it would barely grab the can from the top. Sometimes the claw would even drop our can because our arm didn't go to the best location to pick it up. If we had more data, we would've had a more accurate equation that would've had less error for how we pick up the cans. Another thing we could've improved is where we drop the can after bringing it back. With our design, we could've moved the limit switch further to the left so the forearm goes further back. This would make sure that when our claw opens, it drops the can straight down. With our current design, sometimes the can drops back towards where the claw picked it up from. This has resulted in the can dropped knocking into the other cans. Other than those changes, I would leave the robot as is. We reached the end

goal which was to pick up and move all three cans. We were able to do this consistently. After running our robot multiple times, we were able to pick up at least 2 cans every time and only occasionally would the robot not be able to pick up the third can.

Appendix

```
#pragma config(Sensor, dgtl1,  shaftEncoder1,  sensorQuadEncoder)
#pragma config(Sensor, dgtl3,  change,         sensorTouch)
#pragma config(Sensor, dgtl4,  bumpSensor,     sensorTouch)
#pragma config(Sensor, dgtl5,  limitSwitch,    sensorTouch)
#pragma config(Sensor, dgtl6,  ultSensor,      sensorSONAR_mm)
#pragma config(Sensor, dgtl8,  limitSwitch2,   sensorTouch)
#pragma config(Sensor, dgtl9,  shaftEncoder2,  sensorQuadEncoder)
#pragma config(Motor,  port2,           motor1,        tmotorServoContinuousRotation, openLoop)
#pragma config(Motor,  port3,           motor2,        tmotorServoContinuousRotation, openLoop)
#pragma config(Motor,  port4,           servo,         tmotorServoStandard, openLoop)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

//MOTOR VALUES:
//  -claw servo:
//          Open(during can finding): -70
//          Closed(grabbing can): 100
//  -Motors:
//      Bicep: 50, Negative up, positive down
//    Forearm: 30, negative up, positive down



task main()
{
    int times = 0;
    while (times < 3){ //number of times while loop runs, 1 time for each can
    SensorValue[shaftEncoder1] = 0; //set shaft encoder values to 0
    SensorValue[shaftEncoder2] = 0;
    int dist = SensorValue[ultSensor]; //ultrasonic value which is how far can is from
                                       //the ultrasonic sensor

    //DOWNWARD MOTION OF BICEP
    bool down = true;
    int a = dist*24.72 - 3574.6; //equation generated from linear regression for shaftencoder2
    int b = dist*0.4074 - 229.03; //equation generated from linear regression for shaftencoder1
    motor[servo] = -70; //claw set to open
    motor[motor1] = 30; //move bicep towards tje can
    motor[motor2] = 0;
    //wait1Msec(10000);
```

```
    while(SensorValue[shaftEncoder2] > -a){ //move bicep until it gets to set position
                                            //based on shaft encoder value from linear regression equation
        wait1Msec(10);
    }

    //STOP BICEP AND MOVE FOREARM DOWN
    motor[motor1] = 0; //stop moving bicep, start moving the forearm
    motor[motor2] = 30;
    while(SensorValue[shaftEncoder1] < -b){ //move forearm until it gets to set position
                                            //based on shaft encoder value from linear regression equation
        wait1Msec(10);
    }

    //STOP FOREARM AND PICK UP CAN
    motor[motor2] = 0; //stop moving forearm
  motor[servo] = 100; //close claw to hold can
  down = false;


  //MOVE BICEP UP
  motor[motor1] = -30; //start moving bicep back towards the limit switch
    motor[motor2] = 0;

while (true){ //if bump sensor is hit, move bicep back towards the top/limit switch
    int bump = SensorValue[bumpSensor];
  int limit = SensorValue[limitSwitch];
  int changeDirection = SensorValue[change];
    if (bump == 1){
    motor[motor1]= -30;
    motor[motor2]= 0;
    wait1Msec(5000);
    break;
}
```

```
    //DROP CAN BEHIND ROBOT
    if (limit == 1 && down==false){ //if limit switch is hit, stop moving bicep
        //Bicep motion limit reached
        motor[motor1]=0;
        motor[motor2]=-30;
        //Move forearm back behind elbow/base
        while(SensorValue[limitSwitch2] == 0){ //forearm moves back until other
                                               //limit switch is hit
            wait1Msec(1);
        }
        motor[motor2] = 0; //stop moving forearm

        //release can once forearm limit reached
        motor[servo] = -70; //open the claw
      wait1Msec(1000); //pause for one second
        break;
    }
    if (changeDirection==1){ //code not used
        motor[motor1]=motor[motor1] * -1;
        motor[motor2]=motor[motor2] * -1;
        }
    }
    times = times+1;
    down = true;

}
}
```