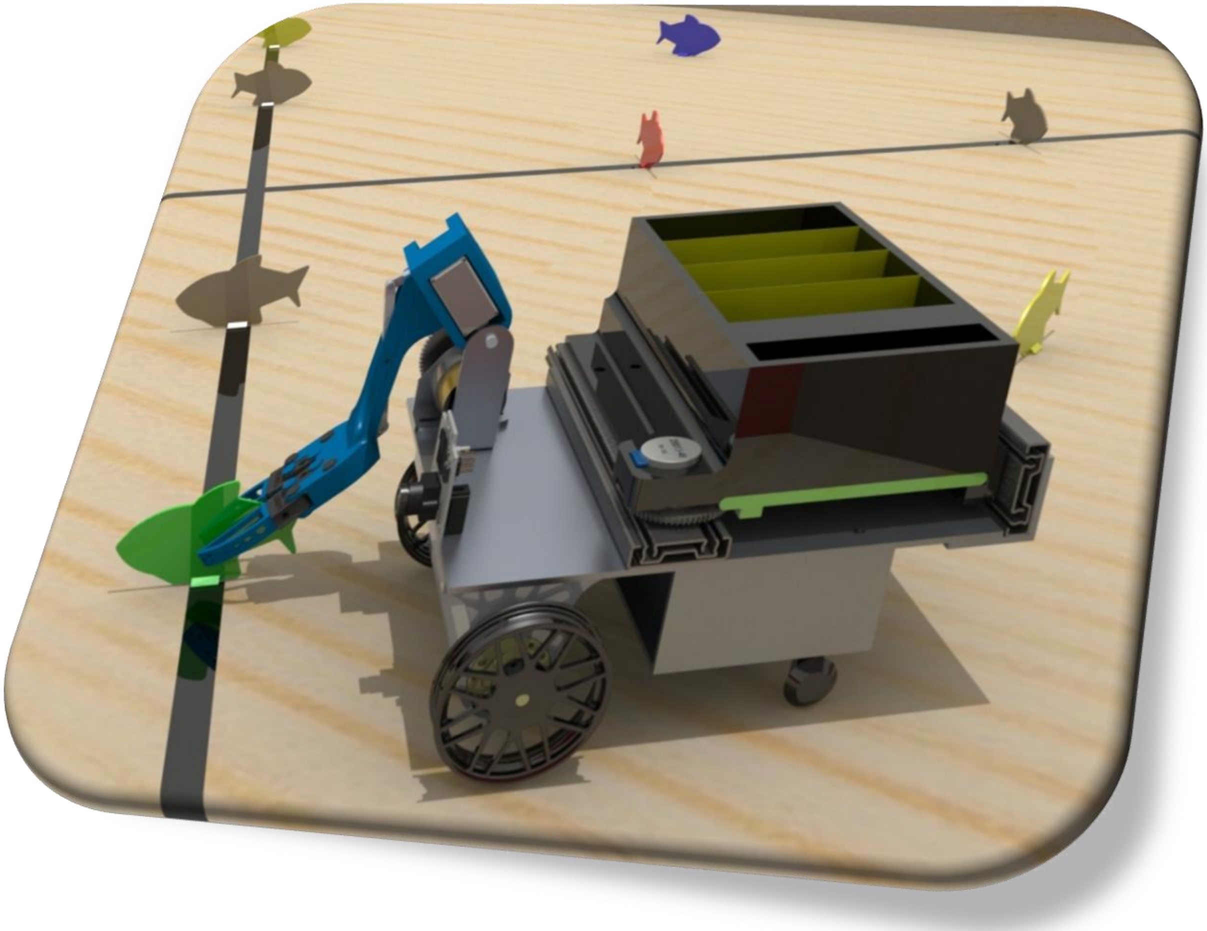
VERSION [1.0]

MAY 22, 2015



ENGINEERING 259 - CAPSTONE PROJECT

The Deadliest Catch

DAVID PRINDLE | BRIAN YOUNGMAN | SUHAIL PRASATHONG | DEREK WELKER

EXECUTIVE SUMMARY

This year’s design competition was fairly rigorous. The problem we were asked to solve was for us to collect fish from across a board that are randomly placed based on color. Once we have picked up these fish, we were to sort them onboard the robot and finally dump then in the bins on the board respective of color. This competition was designed to challenge engineering students at the two-year level where students have the opportunity to develop a solution and implement it with the expertise of up to 4 engineers in their team.

Our team attempted to approach this problem with a rather innovative take. We decided to host a camera onboard the “Deadliest Catch” which would go on to track, mark and label the fish and store it into memory. An effective claw was designed to pick up the fish and dump them into bins that would sort these fish respective of their color. The fish in the bin would then be dropped in bins respectively through a release from under the bins. This design came to us through a significant amount of evolution from the first attempt.

The design came to our team fairly comfortably. Each member of the team very clearly understood all the problems and challenges that would come with the design that we chose. After the physical design was given to us, we worked on the electrical and software components of the robot. We built loops that would repeatedly ensure that the fish were seen and stored with the right tag labels in the cameras memory. Next, we built various code blocks to complete the other tasks such as picking up the fish, sorting and so forth. We ran into various unexpected problems that we never would have predicted. Regardless of limiting the cameras view, it still saw other objects and got distracted. The rest of the functions work with great reliability which was part of the success we celebrated. Overall, we feel that our robot is still a work in progress and has potential to be fully functioning.

TABLE OF CONTENTS

DESIGN EVOLUTION

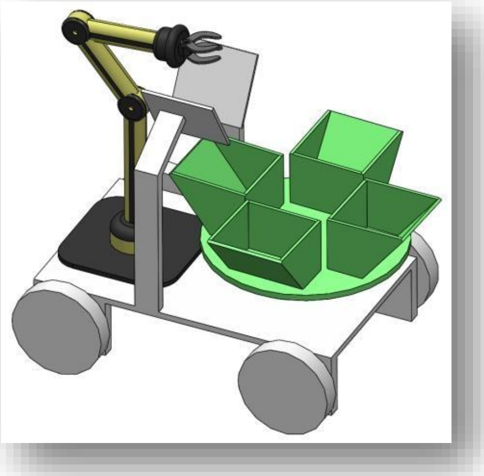
After the four of us formed the group we each came together with differing ideas and concepts. Before we began our first meeting we all made a list of components to use along with ideas for each aspect of the design. These aspects would be the pickup, sorting, storing and drop off mechanisms. A robot as a whole could not be conceived due to the complexity. Almost like one person or group cannot sit down and design an entire car right from the get go. We had to discuss what mechanism we wanted for each aspect and then incorporate that idea into the robot as a whole.

We knew the pickup mechanism would be a claw or something that functioned very similar. For the color sorting we were not sure at the moment if we were going to use color sensors or something else. Determining how you are going to sort the colors greatly affects what mechanically needs to happen. If we were to use a color sensor it would have to be either on the claw or pass through a color sorting operation. We had to take this all into consideration.

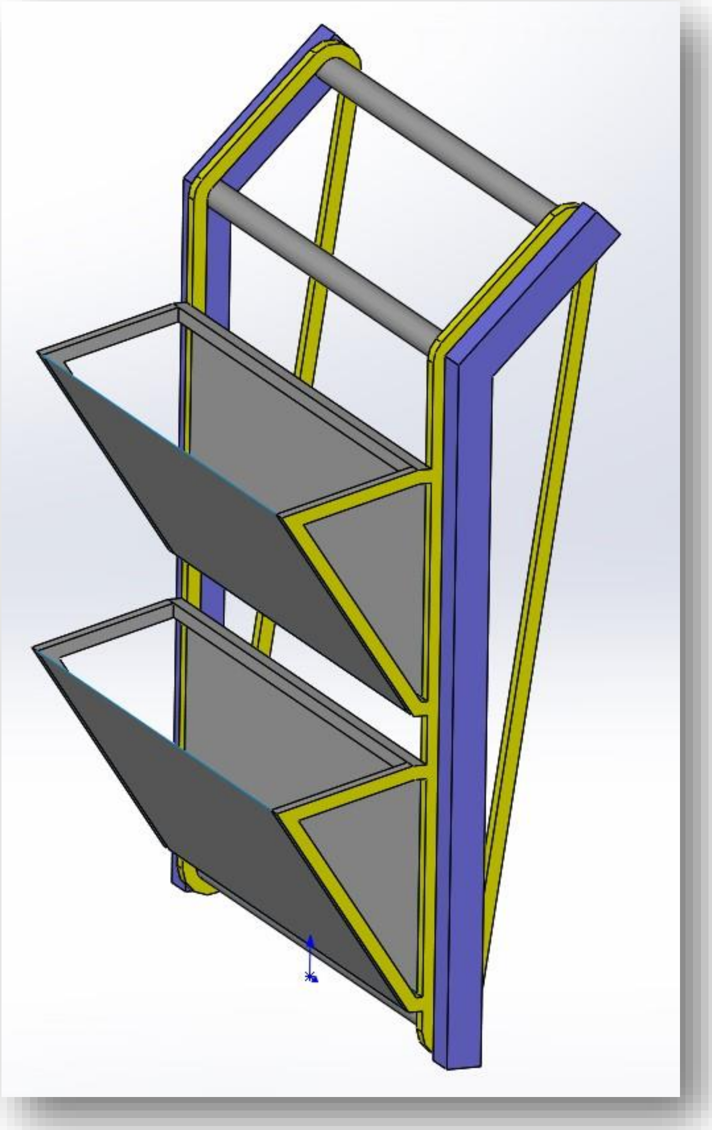
For the sorting and storing aspects we originally decided to have a small "station" for each color. These stations resided on a circular disc that would rotate so the claw could drop a fish into the specified color. After doing some research, we found a Kick Starter project that recently came on to the market that would work perfectly for this task, the PIXY camera system (more information on the PIXY can be found in the electronics' section). The camera can determine the color without making contact from a distance of over 2 meters. This meant that we wouldn't have to incorporate a color sensor into our design, and it would increase the accuracy of color detection.

The drop off mechanism was very dependent on how we decided to store the fish. This is something we actually put off until we had a good grasp on the storing mechanism. Since we didn't know how we would be storing the fish we didn't know exactly how we were going to transfer the fish from the storage bins to the buckets on the track.

Once we had a general concept of how we wanted to approach the ASEE Challenge, we used SOLIDWORKS to draft up a 3D design. SOLIDWORKS enabled us to see a scaled version of the potential robot and rapidly prototype a virtual robot without having to make spend time building and fabricating multiple robot iterations.

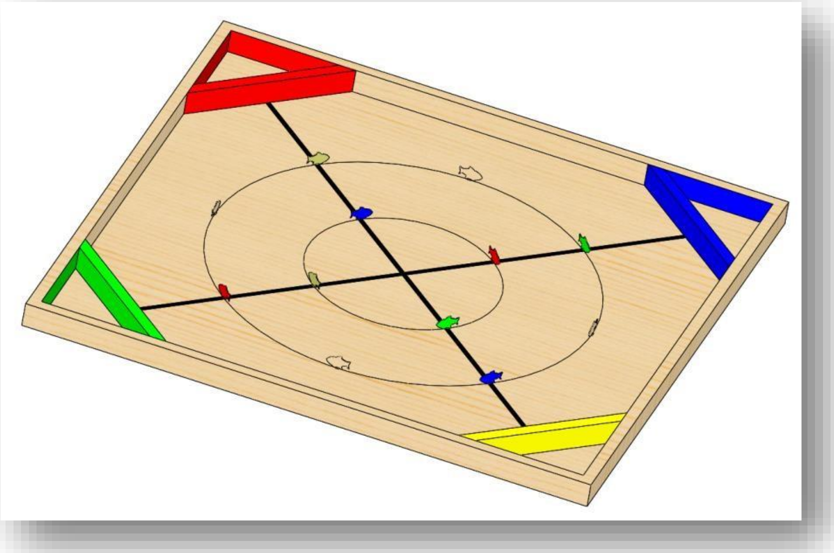


For our first iteration, we initially grabbed a robotic claw from GrabCAD.com for a graphical representation. At this point we did not know if we were going to build our own claw or use an off the shelf component. In the picture above the green boxes and circular disc is our storage mechanism. The disc would rotate around so that the claw would drop the fish into the bucket designated for each specific color. Once we decided that we would sort and store the fish on the go as we collected all of the fish before dropping them off, we began to brainstorm ideas for the drop off mechanism. Initially, we thought that if we put hinges on the bottom edge of the box we could just flip the fish into the buckets. This wasn't a bad idea but we determined it would be difficult to implement. Additionally, we mocked up placing 3 fish into each bucket of the storage system in SOLIDWORKS and realized that it would be difficult to get all of the fish to fit while remaining within the size constraints.



For our first revision we made a new storage and drop off mechanism. As you can see below we placed the robot onto a platform. This platform is the size that we need to stay within. We wanted to incorporate the constraints in as early as possible to rule out if a design was going to be impossible. We also moved the claw over to the side so that it could drop the fish perfectly into the designated spots. Our strategy to pick up the fish was to approach the fish head on so when we picked the fish up it would be collinear to the arm, same direction as the bucket.

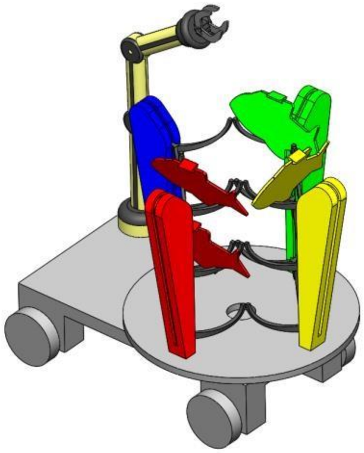
In the picture to the left we have just two spaces and we need to grab three fish. We planed for a third fish cradle that would lie on the angled portion at the top. The yellow portion of the model is a pulley system that would spin the buckets up and over the top.



Our goal was to start the bottom bucket at the top with the other two hanging off the back (to the right side of the picture). When the claw dropped the first fish into the cradle, the system would lower into the position you see so that the next fish can be placed into the bucket. Once all the cradles are all filled, the system would be ready to dispense and a motor would drive the buckets up and over the apparatus letting gravity drop them into the specified dump zones. We played around with the idea of having one large cradle instead of three individual ones but with the limited amount of space we had, and from mock ups in SOLIDWORKS, it just wasn't going to work.

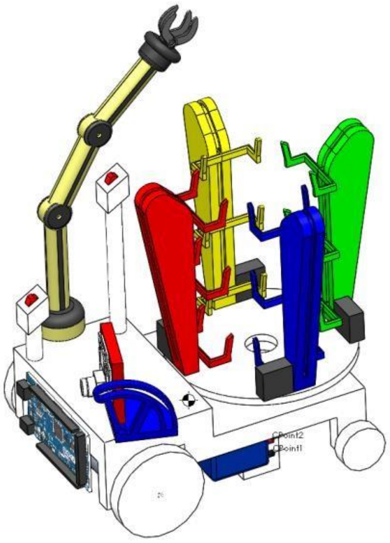
We also decided to model up the entire track to have a good idea how big our robot was going to be. This proved to be very valuable as it allowed us to see if the size of our robot would interfere with other fish when trying to pick them up. We did this very early into the brainstorming process to make sure the size would be acceptable before wasting valuable time drafting and fabricating only to find that the robot would not function in the way in which it was intended.

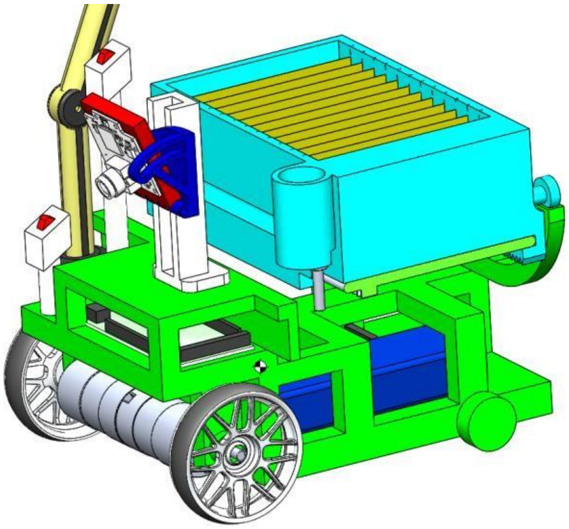
After our first revision we refined the design further. Keeping the same concept, we changed it slightly in order to increase the space for the fish to be stored. The rotating base is exactly the same along with the buckets for each fish but we just changed how they looked so that the buckets would still hold the fish and be within the size requirements. We modeled a groove all the way around the vertical holder which will allow the arms holding the fish to be moved up and down like we discussed in the last revision but reducing the clutter of the system.



In our next revision we altered the arms that would cradle the fish. You can see the difference between the arms from the one on the left compared to the ones on the right. The reason for this is the arms were very close to touching at the ends and they did not need to be that big. We made them smaller so the cradle mechanism would have more room to operate.

In the revision to the left we have added the PIXY camera along with a designed bracket, Arduino Mega board, switches and the battery. The PIXY camera is mounted to a bracket that would be able to adjust to multiple angles as we did not know what angle the PIXY will be best suited at. We wanted the Mega board to be easily accessible so we “mounted” it to the front.





In the latest revision we determined that the large rotating disc was going to be overly complicated to code along with rotating wires and pulleys would prove to be a very difficult mechanical challenge. We scratched that idea and chose a simple concept.

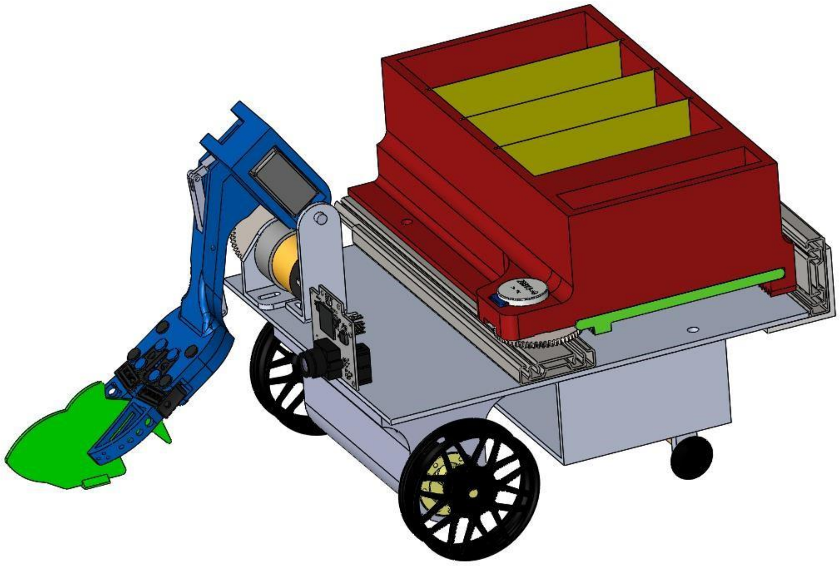
In this design we changed the storing mechanism to a large bucket system. In this bucket, there are slots for every fish. When

the claw grabs a fish the bucket would slide horizontally. Each section within the container is linked to a specific color. For example all the red fish will be right next to each other, so on and so forth.

The bucket has a geared spline so a motor with a gear would be able to drive it back and forth. In the picture to the left there is a component that is red. This is the bottom slide tray for the bucket. It also has teeth in the bottom of it so a motor with a gear could drive it as well. The motor mount would be molded into the bucket inorder to allow it to move with the system.

When the colored fish are ready to be dropped off, the bucket would slide over the drop off section and the bottom plate would then be pulled back. This removes the base from the container and allows the fish to fall out of the bottom due to gravity. The bottom plate is only slid out enough to drop the each color into their corresponding bins.

In the model below, you can see that we added some guides for the container to slide, moved the Arduino board and battery and also added in some drive motor with modeled wheels. At this point, the robot began taking shape.

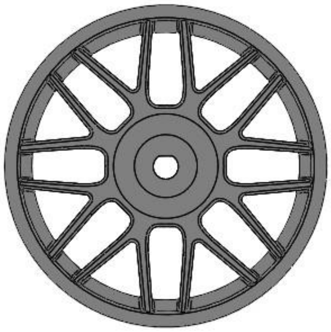


As you can see in the model above we have a similar looking robot as before but almost everything is finalized. We dropped the idea of 3D printing a frame and went back to basics with just a flat metal plate. We also added in the final arm which we designed in SOLIDWORKS. The arm will be discussed in detail below.

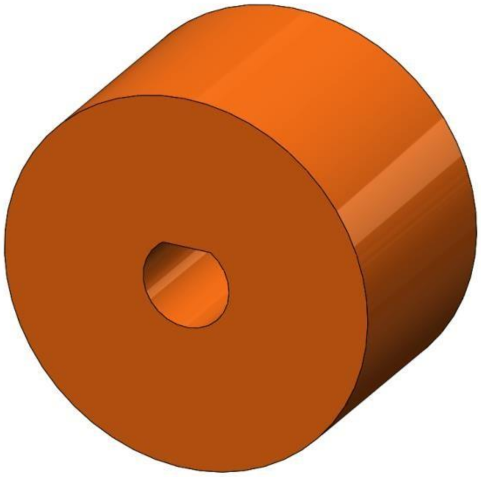
In the back, underneath the flat metal plate we have added a section for the electronics. Also, instead of having a bearing that the storage container rides on we utilized drawer slides. The slides move more fluidly than just a bearing and provided a quick and proven off the shelf solution. The drawer slides would also be able to support the weight of the container when it is extended off the side of the robot when dropping the fish into the track's bins. The only unknown to this model was the PIXY camera. We did not know the best place to mount the camera in order for it to function with the greatest efficiency so we gave it plenty of room on the top to be moved around into the most useful location.

Once the final SOLIDWORKS iteration was complete, the parts that needed to be 3D printed were sent off to CADimensions to be printed and all other parts were fabricated in MCC's machine shop in 9-156. Once completed, there were several changes that evolved. These were unforeseen due to the integration of electronics.

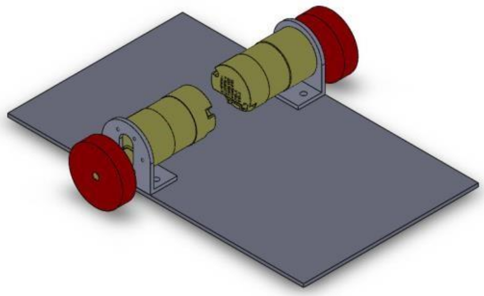
Front Wheels:



In our initial design, we modeled three inch diameter wheels that would be printed on our 3d printer at CADimensions. We opted to print the wheels for several reasons. First, 3D printing is not only simple but also cheap, and quick to do. 3D printing allowed for rapid prototyping which is what we needed and resulted in a professional finish that was much lighter than turning the part down on a lathe and mill. In the wheel design, we included a flat for the shaft of the motor which allowed us to simply press fit the wheels on rather than having to use a set screw.



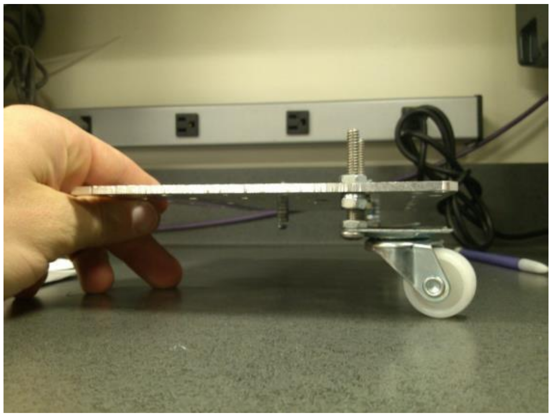
After testing the wheels on the robot, we were not able to get our drive motors to operate with less than an 80 Pulse Width Modulation. We quickly realized that the robot moved too fast for what we were trying to accomplish. This left us with one of three options; to use smaller motors and rebuilt the motor mounts, create a gear box that would gear down our current motors, or simply use wheels that had a smaller diameter. We opted for the later since it was simple, quick and cheap. The new wheels we made were also printed but this time they only had a one inch diameter and no complex design. This resulted in a print time of less than 30 minutes from start to finish. Even though the wheel is very simple this shows the power of 3D printing since in order to mill or turn down this part on a lathe, between setup, fabrication and cleanup time you would these parts would easily take twice the amount of time to create.



The small orange wheels worked well and proved that we could make our robot operate at a slower drive rate however; they did have one major flaw. Since they were so small, the lower chassis plate and motor mounts had to be mounted in an orientation that did not leave a lot of room for the massive amount of electronics to power the robot. Not wanting to waist any more time with designing and fabrication we were able to recycle some old robot parts from an ENR 153 robot. We used the wheels and hubs from the ENR 153 robot that were two inches in diameter. This allowed us to orient the lower chassis plate almost on the ground providing ample room for the electronics while still allowing the robot to travel at slow speeds. Rear Wheels:

Similar to the front wheels, our rear wheel setup evolved as the front wheels did. In our initial design, the rear wheels were comprised of a 3D printed axel support that held a frictionless glide. This rear wheel system was then mounted to the lower chassis plate. There were several flaws with this system, the largest of which was the robots inability to smoothly turn and rotate. The glides moved very easily, but since they only rotated about the axel and were fixed to the chassis plate, they would frequently get caught on the wood grain and imperfections in the wood of the track. This resulted in the robot "catching" as it turned. Another major flaw was that the wheels lacked height adjustment and our robot did not sit perfectly flat on the track. This often resulted in the drive wheels loosing traction because they simply were not making contact with the ground. These inconsistencies would not work for what we set out to accomplish.

We applied a quick fix while testing one day and it seemed to work well. We used a large two inch caser and attached it to the rear chassis plate. The robot was able to make excellent turns and the drive wheels always had solid contact with the track. The large two inch caster did put us outside the size limitations for the competition and was very unstable when our bucket system was extended resulting in the robot tipping over on its side.

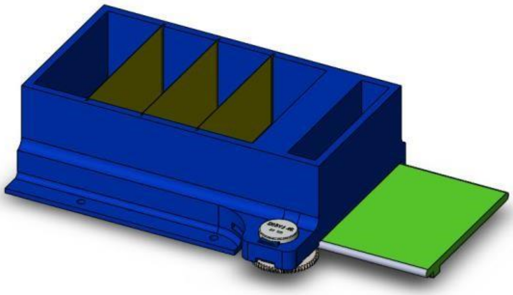


Using the caster concept, we found 1 inch mini casters that would fit under the chassis plate. The caster wheels were mounted to bolts allowing for height adjustment, but proved to be subpar on the track. The mini casters were not of the greatest quality and had a lot of play in them. We were not happy with the results and it was difficult to fit all the electronics on the robot resulting in one last redesign.

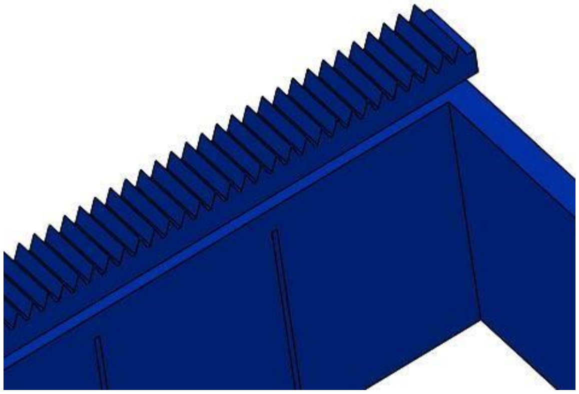
In an attempt to provide more room for the electronics, we used two inch front wheels that allowed us to orient the lower chassis plate almost on the ground and to use a simple furniture glide from Home Depot on the rear on the lower chassis plate. The glide works perfectly. The robot is stable, rotates smoothly, and allowed for plenty of room for the electronics.

Bucket System:

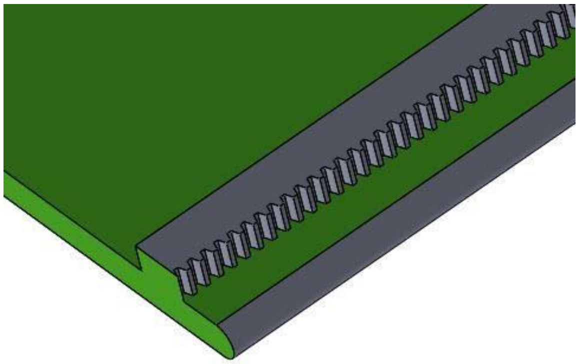
The Bucket System was designed for two things in mind. First, to be able to store each color of fish in their own separate bins and second, to be able to drop off each colored fish group separately. The Bucket System is a box that has four separate compartments for each color group off fish.



The box has no top or bottom. On the bottom of the box is a tray that slides back and forth and is propelled by a stepper motor with a gear and a built in spline of gears along the edge of the tray.



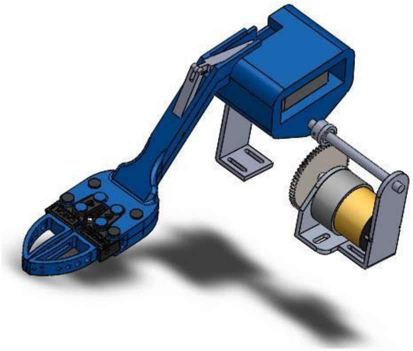
Bucket



Tray Slide

On one of the bottom edges of the box there is also a spline of gears that allows the box to slide back and forth by a stepper and a gear as well. The entire bucket system itself is mounted to a set of 8” drawer slides allowing the box to slide 8” past the edge of the robot. The tray or “bottom” of the box then slides out, allowing the fish to fall out and be dumped into the fishes corresponding colored corner on the track.

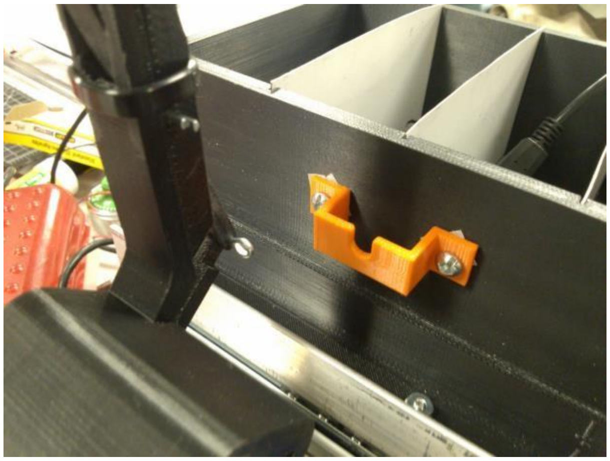
The Claw:



In our initial mock up designs, we used a triple jointed claw from GrabCad.com. This claw would have worked great due to it's ability to move and rotate in almost any direction and configuration but, fabrication and coding of this part would have been just as complex as building the rest of the robot. We needed a claw that would grab the fish and rotate it backwards to place the fish in the bucket. The fish had to be placed in the orientation below in order to fit all twelve fish and the claw system and fit within the size constraints. There were several challenges in creating the claw. Not only did it have to pick up the fish and orient them, but the claw system had to have precise control for coding. We achieved this by rotating the claw about an axel shaft controlled by a stepper motor. In order for the claw to rotate it had to be light. This meant that we could not place any servos at the point in which it grabs because that would mean the mass would be at the farthest point from the fulcrum point. This would require a very substantial motor to rotate
the system and we were limited by our 7.8 Volt system. For more information on the stepper motor we used, please see the electronics section of this report.

Initially we used another model from GrabCad.com. This model was then modified to fit our size requirements as initially the model was over nine inched long and did not include an arm. The GrabCad model also did not include an area or a mount for a motor to actuate the grabbing mechanism. We designed a system of levers to actuate the grabbing mechanism controlled by a servo motor. The Servo motor was mounted within the Claw system itself with its mass centered over the axel of rotation. Built into the System was a hole for a set screw that would allow the claw to be mounted to the shaft. On the shaft we milled two flats, one for the arm and one for a gear. This allowed for a stepper motor to be mounted to the top chassis plate and to spin and rotate the entire system. Later on, we added a pole to one of the claws grippers in order to turn the fish and allow us to pick them up from a head on approach.

For the claw, there was some assembly required. Once assembled, the system worked great. The 3D printed material provided enough structural support without adding much weight as the majority of the mass of the claw is in the hardware. We had to tap 4 holes for the servo motor to bolt up to and one for the set screw using a 6-32 UNC tap.



The bucked system once printed required a little sanding on the lower slide plate, as well as four 632 UNC holes taped; two for the stepper motor and two for the vibration motor bracket on the box. The vibration motor was added in order to shake the fish free as testing proved that they had a tendency to get caught on each other.

ROBOT OPERATION

The robot starts in the middle facing the fish in between the tanks. It will then approach the first fish, adjusting its distance and position to the fish based on the information from the camera. The robot will then lower the pickup arm, and drive to the fish for pickup. Before picking up the fish, the bucket will move to the bin assigned to the current color of the fish. The extender piece on the arm will push the fish to line it up for the claw as the robot moves forward a set distance based on the values given to the encoders. When the robot is in position for pickup, the robot will stop and grab the fish with the claw. After a short delay, the pickup arm will lift the fish back and drop it into the bucket. After the first fish, the robot will attempt to repeat the same steps for the second fish. With the two fish in between the tanks gone, the robot will drive in reverse a set distance based on the value given to the encoders, and turn towards the fish in front of the bucket. The robot will then approach the fish as it did the previous two, and attempt a pickup. With the fish in front of the bucket gone, the camera will have a good view of the tank and can use it to reverse back to the center. Once the robot is back to the center, it will turn towards the next set of fish in between the tanks, and repeat the process all over again. This process will be done a total of 4 times to capture all fish, at which point the robot will be ready for drop off. Because of the way the bucket system works, the robot must go to the tank that corresponds with the color of the fish in the first bucket, which it will be at after the last fish is picked up. Already at the tank for the first drop off, the robot will get as close to the wall as it can without touching it, and then rotate in place. The bucket will with then be extended out far enough to reach over the tank, and the floor will be moved out to let the fish drop. The robot will then rotate around so the camera is facing the tank it just dropped off too, and back up till it is back to the center. Using the camera to find the tank corresponding to the color of the next bucket, the robot will approach the tank as it did the previous tank. This process will be repeated 2 more times and all fish will be sorted and dropped off.

ELECTRONICS AND SOFTWARE

Introduction to Electronics

The electronics are a key component to the robot. We believe that at the heart of the robot are the electronics components that breathe life into the robot. Our team was able to bring together an intricate set of electrical components that aided in our design and robot operation. The various components brought together in unison allowed for a smoothly operating robot, however, we clearly had trouble meeting the said requirements. Nevertheless, our robot is still a work in progress and is expected to function fully within the next week.

Components Breakdown

All the electrical components onboard The Deadliest Catch are described as follows:

The Microcontroller

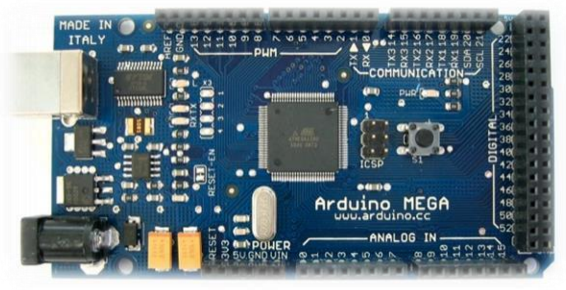


Figure 3.0

Our team decided to implement the Arduino Mega microcontroller which turned out to be exactly what we needed as far as number of ports, h-bridge installations and onboard equipment that was needed. The Mega consists of 54 input pins, 16 MHz clock speed, 8 KB of onboard SRAM and a traditional 5V operating voltage really served us with our needs.

The Batteries

We decided to bring on two batteries on the Deadliest Catch. The 1000 mAh Kingmax Lipo battery we used was simply to power the Arduino. The sole reason we decided to separate the battery sourcing is because we did not want to have any interference in power signals. Generally, we found that this was an extremely efficient decision and was a great counter measure to ensure that there is no interference.

Figure 3.1

The other batter we implemented was Duratrax Lipo Onyx 5000mAh. We used this for practically everything else that needed power on our robot. Separating the power source into two sources was probably one of the smartest decisions we made on this robot.

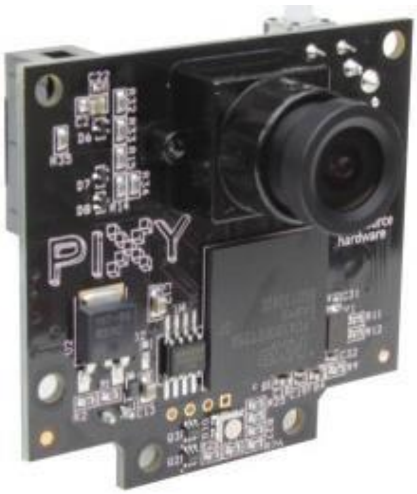
Figure 3.2

The Switch

The switch, although the most basic component of the robot that hardly costs a dollar probably holds one of the most significant responsibilities for the robot. We were able to utilize the switch to have it instantly autocorrect, focus the lens and start the robot operation as soon as the-flip is switched on.

Figure 3.3

The PIXY Cam



The PIXY Camera was definitely one of the most innovative ideas we were able to incorporate into the robot. The PIXY essentially views a certain object, sets a size for it and simply labels it for recall in the future. The PIXY has proven to have good results, moreover, we found that the PIXY has been extremely consistent with the fish and any other test prospects that we have attempted to scan. The development of the PIXY product is pretty fresh and has a few bugs with libraries but we were able to work around that to make it function to our needs.

Figure 3.4

The PIXY Cam

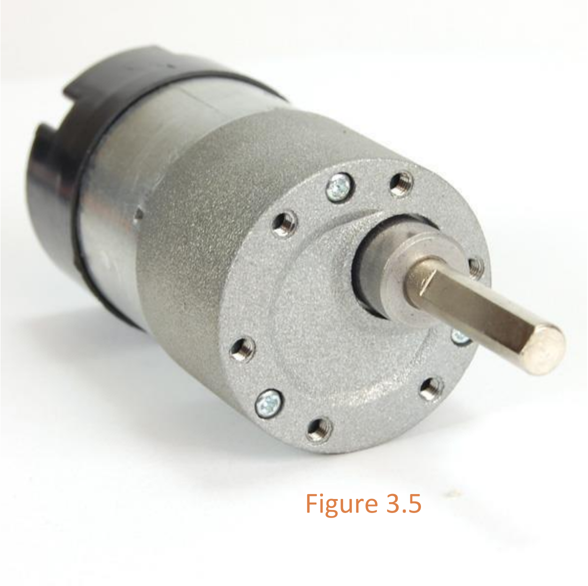


Figure 3.5

Our motors were Trossen Robotics DC gearhead motors with a 6V pulsation (180rpm) with encoders onboard the gear. The gear system was effectively put in place and hooked to the power source respectively. We did not find too many problems with gears at all and it was a fairly simple affair to implement.

Breadboard Wires

Although this might seem like a minor component for the monstrous robot that we have at hand, it truly is what holds all the electrical components together. These wires essentially serve as nerves for our robot.

Figure 3.6

Stepper Motors

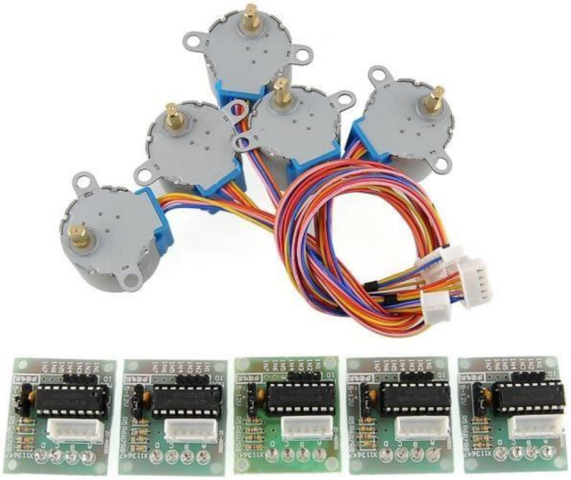
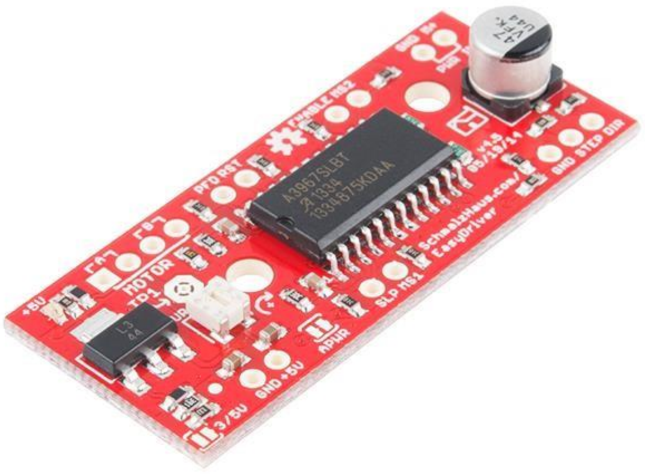


Figure 3.7

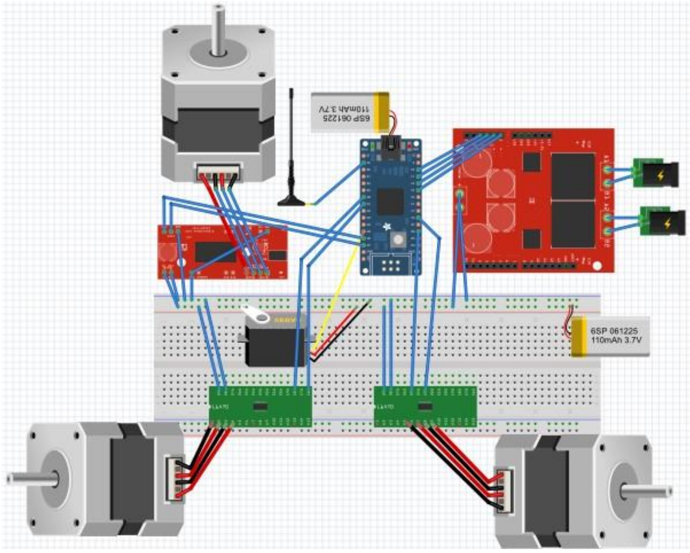
The stepper motors played a huge role in our motor operations on the Deadliest Catch. With a 5V DC 4-Phase 5 Wire Stepper, we were able to successfully role a mammoth 10 lbs. robot without much backfire. The navigation has been extremely smooth. Our concerns lie more with the code of the navigation system than the electrical components.

Stepper Motor Drivers



The stepper motor driver is essentially what gave all the directions to the stepper motors. Since we had multiple implications of the stepper motors, we needed the motor drivers to push those motors and their encoders in the right direction so to speak.

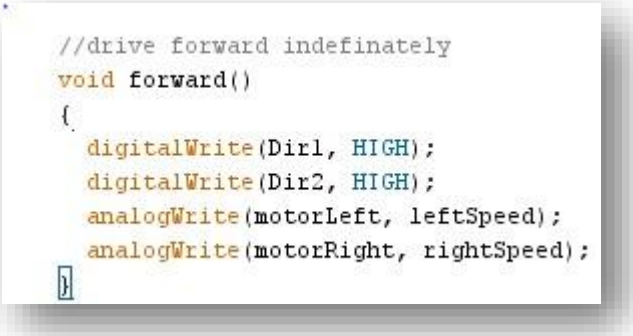
Circuitry



The circuitry on the left clearly represents a lot of the electronic prospects we had onboard the Deadliest Catch. The camera is hooked up as the antenna, the blue block represents the Arduino Mega with all the appropriate ports connected to the motors, vibration system and so forth. All of these components have been discussed in detail in the design evolution portion of this analysis. Please refer to *article 2* in the table of contents to learn more.

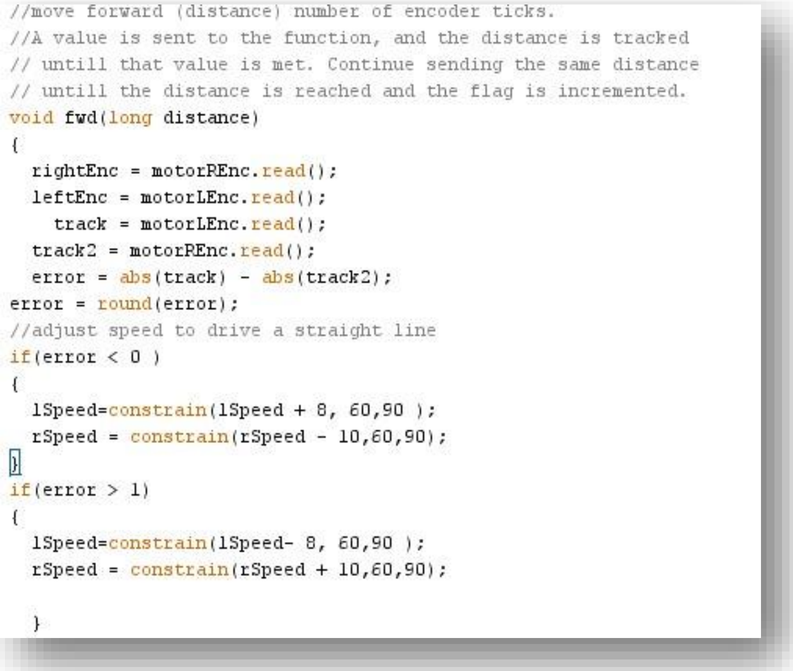
Software Analysis

The forward function drives the robot forward until another command is given for it to stop. All pins are named variables that are initialized at the start of the program to make it easier to change pins. The function starts by setting the direction pins on the motor controller to HIGH, which moves the robot forward. It then writes a speed to the motor controller drive pins. The speed value sent is a PWM(Pulse Width Modulation) value between 0 and 255, which is a way to tell the motor controller to repeatedly turn on and off power to the motors. This results in a different voltages being applied to the motors for different values of PWM, changing the speed of the motors. In practice, values below 60 are too low to drive the motors and values above 150 tend to be too fast to control. This function is used when the robot needs to move forward until the camera indicates the robot has reached a certain distance to the robot.



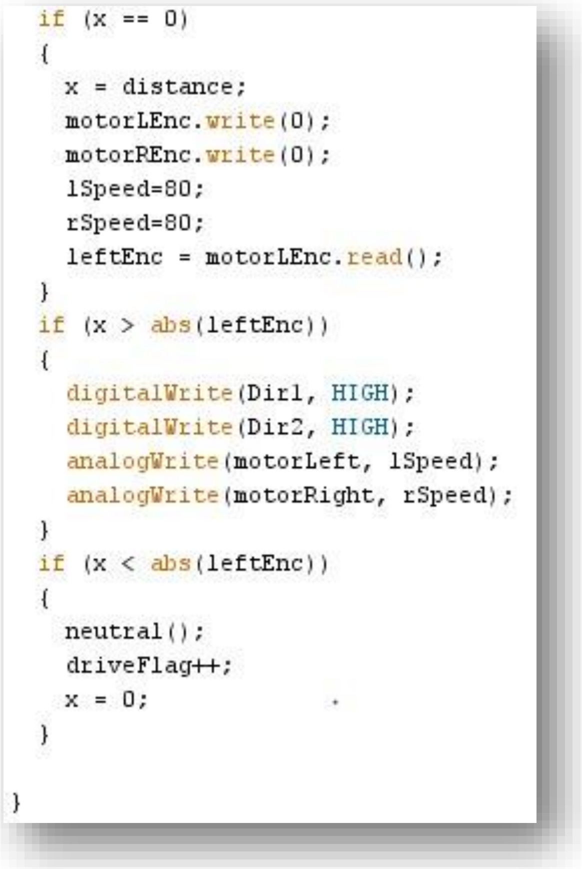
```
//drive forward indefinately
void forward()
{
  digitalWrite(Dir1, HIGH);
  digitalWrite(Dir2, HIGH);
  analogWrite(motorLeft, leftSpeed);
  analogWrite(motorRight, rightSpeed);
}
```

A second function, named “fwd”, was added so the robot could drive forward a set distance while attempting to move in as straight a line as possible. The biggest difference between this function and the one named “forward” is that this function receives a distance value. This value is used to determine how many encoder steps the robot should go. The goal of this function is to be able to move the robot forward, in a straight line, while not requiring the program to stay in the function. To achieve this, it first starts by reading



```
//move forward (distance) number of encoder ticks.
//A value is sent to the function, and the distance is tracked
// untill that value is met. Continue sending the same distance
// untill the distance is reached and the flag is incremented.
void fwd(long distance)
{
  rightEnc = motorREnc.read();
  leftEnc = motorLEnc.read();
    track = motorLEnc.read();
  track2 = motorREnc.read();
  error = abs(track) - abs(track2);
error = round(error);
//adjust speed to drive a straight line
if(error < 0 )
{
  lSpeed=constrain(lSpeed + 8, 60,90 );
  rSpeed = constrain(rSpeed - 10,60,90);
}
if(error > 1)
{
  lSpeed=constrain(lSpeed- 8, 60,90 );
  rSpeed = constrain(rSpeed + 10,60,90);

  }
```

the encoders to determine how far the robot has moved and if the robot has been moving in a straight line. If the encoder values have a difference between them, the motor speeds are adjusted to help compensate. The change in motor speeds are restricted so that one motor doesn’t turn off while another is going very fast in the event there is a problem reading the encoders. The function then checks if the variable ‘x’ has a value or not. This value is the number of encoder steps the robot needs to take. If this value is 0, then ‘x’ will be the distance passed to the function, the encoder values will be cleared, and the motor speeds will be reset to a starting speed. The function then compares ‘x’ to the values read off the encoders, and if the encoders are less than the desired distance(‘x’), the function adjusts the speed as previously determined. If the desired distance has been reached, the function will stop the robot, increment a drive flag to indicate to the rest of the program the distance has been reached, and resets the distance tracking variable ‘x’. The advantage to this method is that as long as the same value is written to the function, and the flag is used to track when the distance has been reached, the program can continue cycling through other functions to gather data or perform other actions.

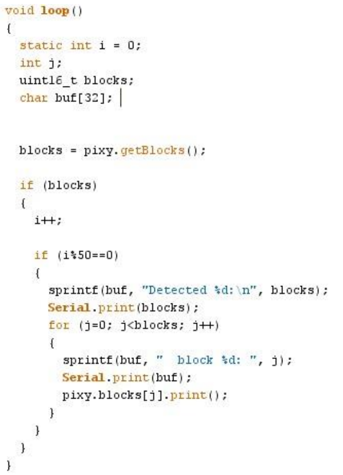


```
  if (x == 0)
  {
    x = distance;
    motorLEnc.write(0);
    motorREnc.write(0);
    lSpeed=80;
    rSpeed=80;
    leftEnc = motorLEnc.read();
  }
  if (x > abs(leftEnc))
  {
    digitalWrite(Dir1, HIGH);
    digitalWrite(Dir2, HIGH);
    analogWrite(motorLeft, lSpeed);
    analogWrite(motorRight, rSpeed);
  }
  if (x < abs(leftEnc))
  {
    neutral();
    driveFlag++;
    x = 0;
  }

}
```

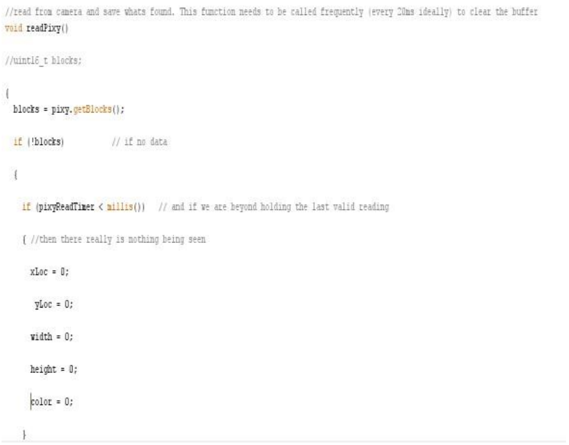
The program uses other functions based on the same principle of the **forward** and **fwd** functions, named **reverse, rvrs, turn, turnLeft and turnRight**. The only changes will be the directions, a LOW to both will drive the robot in reverse, a LOW to one direction and a HIGH to the other will cause the robot to turn.

PIXY Camera

The manufacturers default code made for the pixy was useful for testing what the pixy sees. It starts by initializing variables to track object count, create a buffer of the object data and monitor the number of times the data is printed to the user. It then grabs data from the camera and checks if the camera sent any data, and if it has the Arduino will print the information to the user. It will only print data every 50 times the camera has found object so as not to overwhelm the Arduino processor. The program takes about 20 milliseconds to run, which results in object information being displayed approximately once per second. The benefit of this program in testing was to identify what the camera was seeing, and what data the Arduino was using to act on.



```
void loop()
{
  static int i = 0;
  int j;
  uint16_t blocks;
  char buf[32];


  blocks = pixy.getBlocks();

  if (blocks)
  {
    i++;

    if (i%50==0)
    {
      sprintf(buf, "Detected %d:\n", blocks);
      Serial.print(blocks);
      for (j=0; j<blocks; j++)
      {
        sprintf(buf, "  block %d: ", j);
        Serial.print(buf);
        pixy.blocks[j].print();
      }
    }
  }
}
```

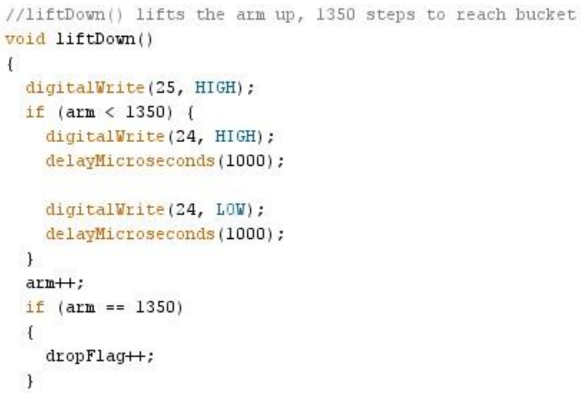


```
//read from camera and save whats found. This function needs to be called frequently (every 20ms ideally) to clear the buffer
void readPixy()

//uint16_t blocks;

{
 blocks = pixy.getBlocks();

 if (!blocks)          // if no data

 {

  if (pixyReadTimer < millis())   // and if we are beyond holding the last valid reading

  { //then there really is nothing being seen

   xLoc = 0;

    yLoc = 0;

   width = 0;

   height = 0;

   color = 0;

  }
```

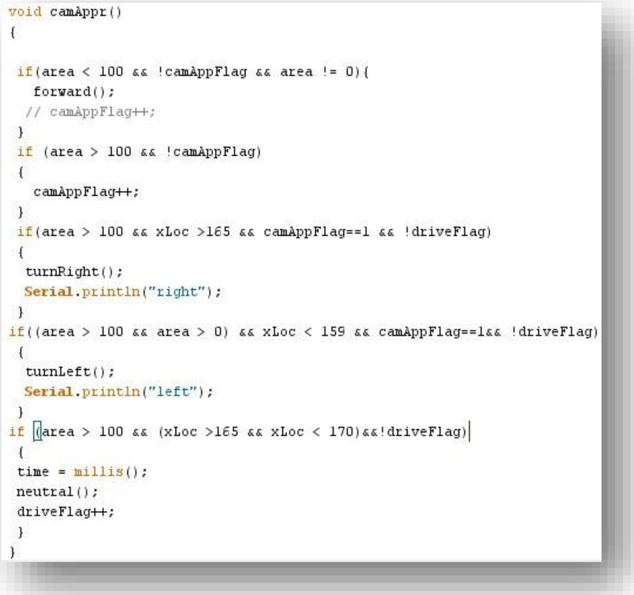
In operation, the pixy requires some tricky coding to be useful. Due to the way the manufacturer wrote the library for the Arduino, the camera will save data till the camera is read. This means the camera needs to be constantly read, and once the data is read, the cameras data is cleared, and will return a zero if called in less than 20 milliseconds because it hasn't been updated. To work around this, the program will cycle through all functions

repeatedly, and the **readPixy** function will only clear data if the camera returns no data twice in 25 milliseconds. If an object is found in the camera, the function will update variables for the color, location and size of the object found.

After the camera has been read, the data needs to be used. This is where the **camAppr** function comes in. When this function is called, the robot will drive towards the fish until the camera indicates the object is a particular size, which means the robot is close enough to attempt pick up. Once the robot is close enough, the function will increment a flag to make sure the robot does not attempt to go any closer. After this flag



```
//liftDown() lifts the arm up, 1350 steps to reach bucket
void liftDown()
{
  digitalWrite(25, HIGH);
  if (arm < 1350) {
    digitalWrite(24, HIGH);
    delayMicroseconds(1000);

    digitalWrite(24, LOW);
    delayMicroseconds(1000);
  }
  arm++;
  if (arm == 1350)
  {
    dropFlag++;
  }
```

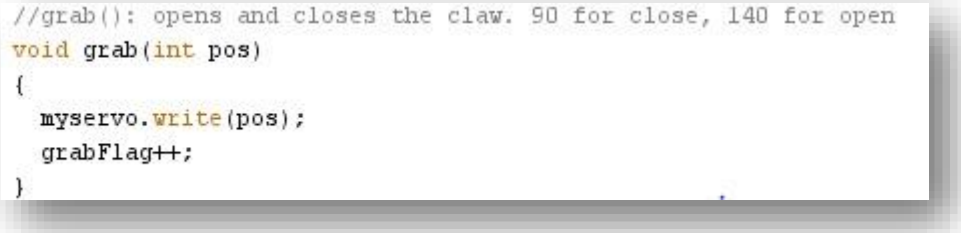


```
void camAppr()
{

 if(area < 100 && !camAppFlag && area != 0){
   forward();
  // camAppFlag++;
 }
 if (area > 100 && !camAppFlag)
 {
   camAppFlag++;
 }
 if(area > 100 && xLoc >165 && camAppFlag==1 && !driveFlag)
 {
  turnRight();
  Serial.println("right");
 }
if((area > 100 && area > 0) && xLoc < 159 && camAppFlag==1&& !driveFlag)
 {
  turnLeft();
  Serial.println("left");
 }
if (area > 100 && (xLoc >165 && xLoc < 170)&&!driveFlag)
 {
 time = millis();
 neutral();
 driveFlag++;
 }
}
```

is raised, robot will then turn to orient itself in such a way that the fish can be picked up. When the robot is in position, the **camAppr** function will raise another flag to let the robot know its time to pick up.

Most of the PIXY Camera code was pretty on point to our expectations but all the problems we ran into were pretty straightforward to deal with and take care of. Overall, we were able to successfully put together the PIXY cam code and it does what we expect it to.

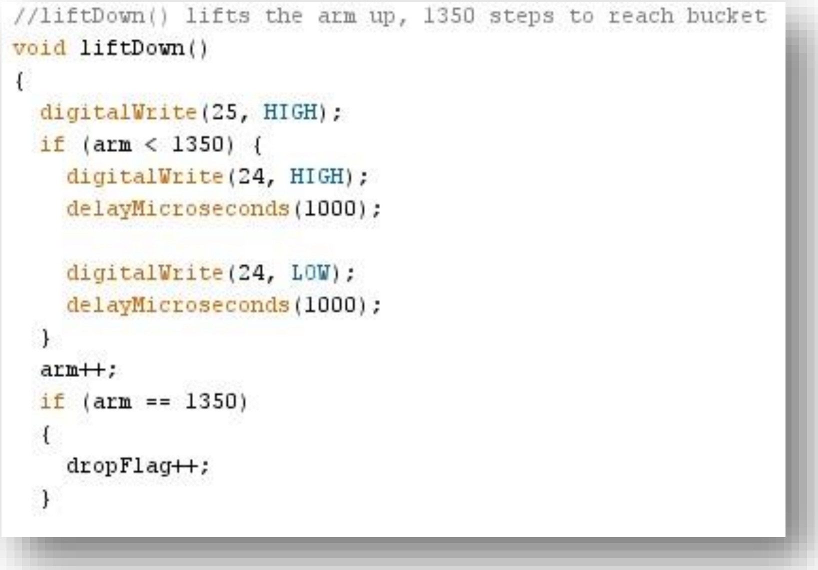
We truly hope to see this robot functioning successfully sooner rather than later. It is important to note that although the PIXY cam does well, there are other components that rely on the PIXY cam that does not work the way we expect it to.



```
//grab(): opens and closes the claw. 90 for close, 140 for open
void grab(int pos)
{
  myservo.write(pos);
  grabFlag++;
}
```

Claw Operation

The claw is operated by a servo, and the Arduino has built in servo libraries. The control of the claw is simply to write a position (0-180) to the servo object. Based on the way the servo is attached to the claw, sending the servo a position of 90 will close the claw, and a position of 140 will open the claw. After the claw is opened or closed, the **grab** function will increment a flag to indicate the next step in the process can be done.



```
//liftDown() lifts the arm up, 1350 steps to reach bucket
void liftDown()
{
  digitalWrite(25, HIGH);
  if (arm < 1350) {
    digitalWrite(24, HIGH);
    delayMicroseconds(1000);

    digitalWrite(24, LOW);
    delayMicroseconds(1000);
  }
  arm++;
  if (arm == 1350)
  {
    dropFlag++;
  }
```

Arm Operations

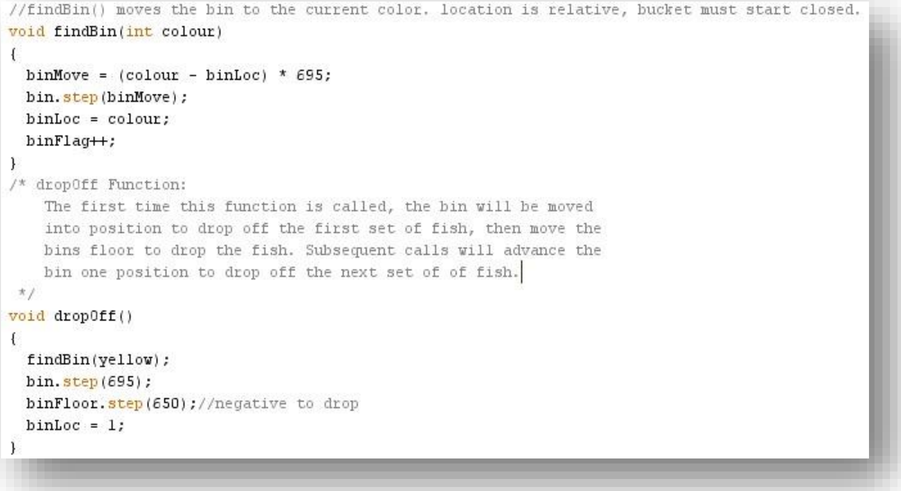
The arm is operated through the **liftUp** and **liftDown** functions. First, the Arduino pin attached to the DIR pin of the motor controller is brought HIGH to lift the arm up towards the bucket, or LOW to drop the arm down for pickup. Then the pin on the Arduino attached to the ENABLE pin on the motor controller is cycled through a HIGH/LOW state with a short delay in between (1000 micro seconds). This will move the arm through one “step”, which is approximately 1.7 degrees. The “arm” variable is then incremented to track the location of the arm. The main program will monitor the location of the arm, and only call the function as many times as needed to reach the desired location(approximately 1350 steps for full rotation).

Bucket Operation

The bucket is controlled by two functions. **findBin** receives the color of the fish picked up, and then determines how far to move the bucket. The stepper motor controlling the bucket uses the built in Arduino library **STEPPER** to move the bucket, which means the stepper will be given a set number of steps to move, and then the library handles the actual movement. Control is lost until the stepper moves through all the steps.

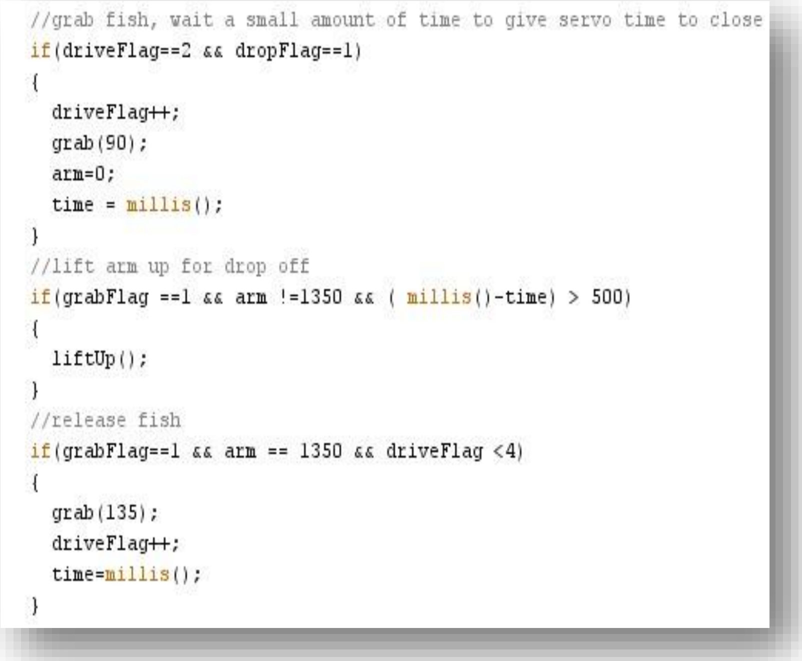
Final Loop

The only indication of the bucket location is based on what the program has tracked, therefore the bucket must always start in a closed position. There are 695 steps between each section of the bucket, and the function determines where to move the bucket by the difference in the current colored bin, and the desired colored bin.



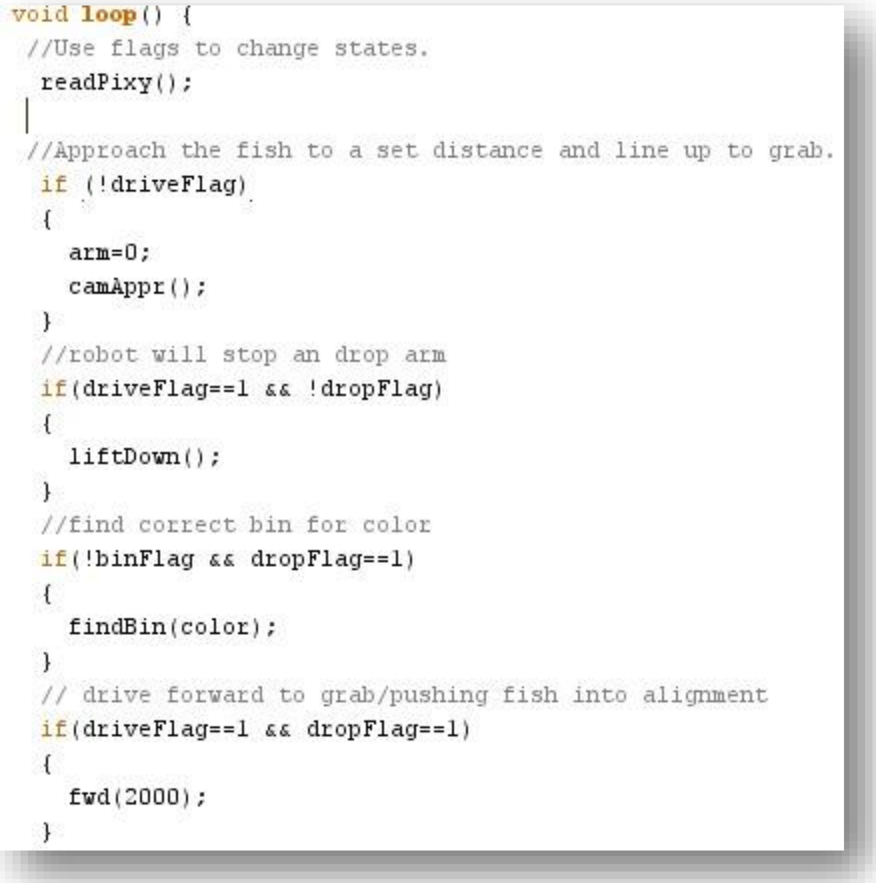
```
//findBin() moves the bin to the current color. location is relative, bucket must start closed.
void findBin(int colour)
{
  binMove = (colour - binLoc) * 695;
  bin.step(binMove);
  binLoc = colour;
  binFlag++;
}
/* dropOff Function:
    The first time this function is called, the bin will be moved
    into position to drop off the first set of fish, then move the
    bins floor to drop the fish. Subsequent calls will advance the
    bin one position to drop off the next set of of fish.
 */
void dropOff()
{
  findBin(yellow);
  bin.step(695);
  binFloor.step(650);//negative to drop
  binLoc = 1;
}
```

A negative value for this difference will result in a negative number of steps, which the stepper library will interpret as a requirement to move the stepper in reverse. After the bin is moved, the variable "binLoc" holds the current color, and a flag is incremented to indicate the bucket has moved. The floor of the bucket, which is moved for drop off, is controlled by the **dropOff** function. First, this function uses the **findBin** function to move the bucket in place for drop off, and then moves the floor to drop off one set of colored fish. Subsequent calls to this function will move the floor to the next set of colored fish. The functions do most of the heavy lifting. Due to the fact the camera needs to be constantly read, the flags from each function



```
//grab fish, wait a small amount of time to give servo time to close
if(driveFlag==2 && dropFlag==1)
{
  driveFlag++;
  grab(90);
  arm=0;
  time = millis();
}
//lift arm up for drop off
if(grabFlag ==1 && arm !=1350 && ( millis()-time) > 500)
{
  liftUp();
}
//release fish
if(grabFlag==1 && arm == 1350 && driveFlag <4)
{
  grab(135);
  driveFlag++;
  time=millis();
}
```

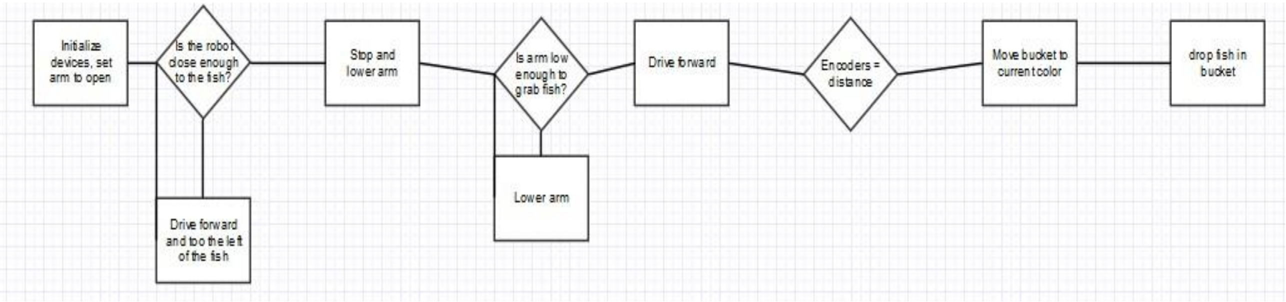
tell the main program what its next step should be. All the functions either take a value to determine when that function has fulfilled its function.



```
void loop() {
 //Use flags to change states.
  readPixy();

 //Approach the fish to a set distance and line up to grab.
  if (!driveFlag)
  {
    arm=0;
    camAppr();
  }
  //robot will stop an drop arm
  if(driveFlag==1 && !dropFlag)
  {
    liftDown();
  }
  //find correct bin for color
  if(!binFlag && dropFlag==1)
  {
    findBin(color);
  }
  // drive forward to grab/pushing fish into alignment
  if(driveFlag==1 && dropFlag==1)
  {
    fwd(2000);
  }
```

At the start, the program will read the camera. It will then go through a series of "if" statements to check where in the operation the program is, and where it needs to be. The first check is if the robot is close enough to the fish for pick up. Then it checks if the robot is close enough, and if the arm has been dropped down. After the arm is dropped down, the program checks the bin location and then checks if the robot is in position to grab the fish. The next part checks if the fish has been grabbed, and will track the time since the fish has been grabbed to delay the arm from lifting up before it has a fish. The "delay" function is not used here because it would cause the cameras buffer to build up. The final checks determine when the arm and the claw are in position to drop the fish off in the bucket.

Software Flowchart & Conclusion


Initialize devices, set arm to open
Is the robot close enough to the fish?
Drive forward and too the left of the fish
Stop and lower arm
Is arm low enough to grab fish?
Lower arm
Drive forward
Encoders = distance
Move bucket to current color
drop fish in bucket

The flowchart above encompasses a very basic representation of the software process that we implemented. It represents the thought process we developed as we built the code. Conclusively, we believe that our software has most of the issues in the robot at this time and a new implementation process for the code is due.

FABRICATION METHODS

Introduction to Fabrication

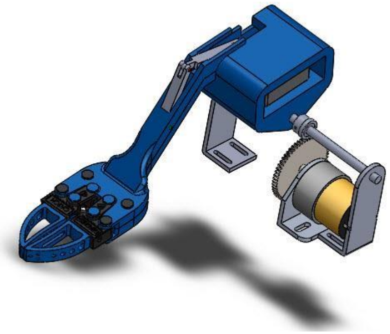
Going into this project, our initial idea was to 3D print the entire design. However due to the size of the robot, the cost of printing material would be too great. Additionally it would severely reduce our ability to make changes as the robots design evolved. So, we combined the use of 3D printed parts and aluminum. For the most complex parts, we had them 3D printed, and for the simpler parts we opted to machine them out of aluminum.

Starting with parts we opted to print, we simply created a .STL file from our SOLIDWORKS designs and sent them to CADimensions to be printed. Both Derek and Dave work there, allowing us to use their printers. CADimensions' printers are of a much higher quality than MCC's 3D printer resulting in not only a better finish, but also a stronger part.

We opted to print our claw, bucket system, vibrator motor bracket, front wheels, and rear axle assembles. The bucket system and arm were printed due to the necessity of a light weight material, but also because of their complexity. As you can see for the drawings below, fabrication of these parts is beyond the scope of MCC's machine shop abilities. Once printed, the parts were placed in an acid bath to dissolve all of the support material. After the material was removed, we just had to tap a few of the parts to allow us to bolt up a few electronics.

The Claw System

Initially we used another model from GrabCad.com. This model was then modified to fit our size requirements as initially the model was over six inches long and did not include an arm. The GrabCad model also did not include an area or a mount for a motor to actuate the grabbing mechanism or the ability to rotate.



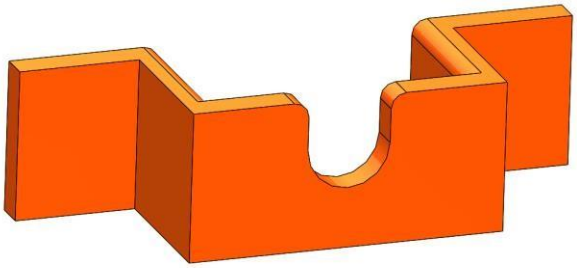
SOLIDWORKS Steps:

The Claw System consists of 46 components not including hardware. The each component was designed in SOLIDWORKS using a combination of extrudes, cuts, sweeps, revolves, mirrors, fillets, patterns, shells, and surfacing. All Part files can be furnished upon requests but to list the specific steps would require an additional report of equal or greater length.

Fabrication Steps:

All part files were converted to STL files and printed on CADimensions Objet 3D printers.

Vibration Motor Bracket



SOLIDWORKS Steps:

Thin Extrude profile
Extrude Cut Holes
Fillet edges

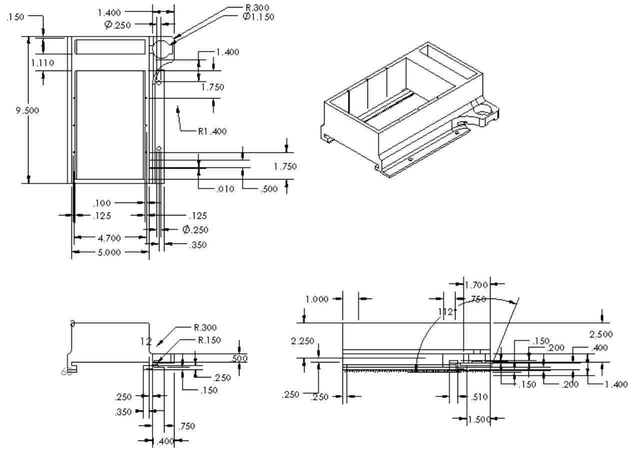
Fabrication Steps:

Part file was converted to .STL file and printed on CADimensions Objet 3D printers.

The Bucket System – Bucket

SOLIDWORKS Steps:

The bucket was designed in SOLIDWORKS using a combination of extrudes, cuts, sweeps, revolves, mirrors, fillets, patterns, shells, and surfacing. All Part files can be furnished upon requests but this part has over 40 features and a step by step process would not be practical.



Fabrication Steps:

Part file was converted to .STL file and printed on CADimensions Objet 3D printers.

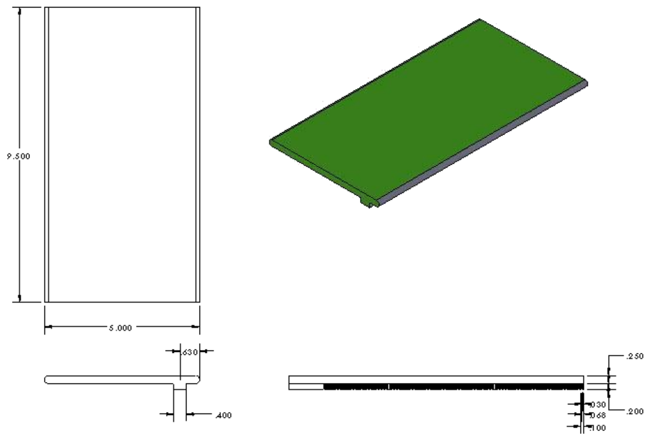
The Bucket System – Slide Tray

SOLIDWORKS Steps:

Boss extrude outer dimension
Thin extrude gear spline
Cut extrude 1 gear tooth
Linear pattern teeth
Fillet edges

Fabrication Steps:

Part file was converted to .STL file and printed on CADimensions Objet 3D printers.



* The front wheels and the rear axle assembly were originally designed in SOLIDWORKS and 3D printed but was later abandoned for off the shelf components.

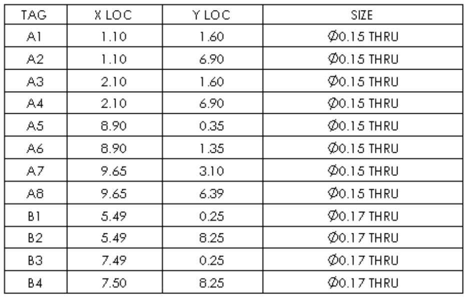
For the rest of the parts we created them in Building 9-156 Fabrication Lab. All Parts were first created in SOLIDWORKS, and then milled and lathed to specs. The following is the speeds and feeds we used for each material used:

All milling done with a 5/16 endmill with RPM=900 IPM=2 unless otherwise noted.
All turning of plastic done with RPM =200-300
All turning of metal done with RPM of
700 Drilling RPM on lathe =200-300
Drilling RPM on mill =200-300.

Top Chassis Plate

SOLIDWORKS Steps:

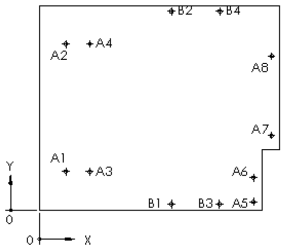
Create a rectangle the correct outside dimensions
Extrude boss it out the correct depth
Extrude cut the holes



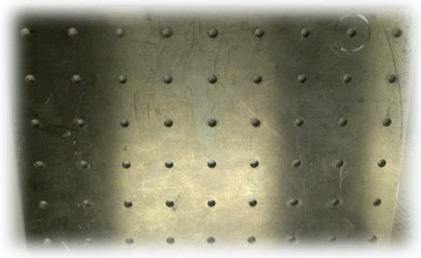
TAG	X LOC	Y LOC	SIZE
A1	1.10	1.60	Ø0.15 THRU
A2	1.10	6.90	Ø0.15 THRU
A3	2.10	1.60	Ø0.15 THRU
A4	2.10	6.90	Ø0.15 THRU
A5	8.90	0.35	Ø0.15 THRU
A6	8.90	1.35	Ø0.15 THRU
A7	9.65	3.10	Ø0.15 THRU
A8	9.65	6.39	Ø0.15 THRU
B1	5.49	0.25	Ø0.17 THRU
B2	5.49	8.25	Ø0.17 THRU
B3	7.49	0.25	Ø0.17 THRU
B4	7.50	8.25	Ø0.17 THRU

Fabrication Steps:

Find zero
Program mill path to take off area around front wheel mount and mill it
Set up a mill-stop
Flip chassis over and mill the other side
Turn chassis around and program mill to fillet rear corner and mill it Flip chassis over and mill other rear corner
Program and mill the back rectangular pocket larger
Program positional drill locations and center drill and drill additional mounting holes.



Lower Chassis Plate

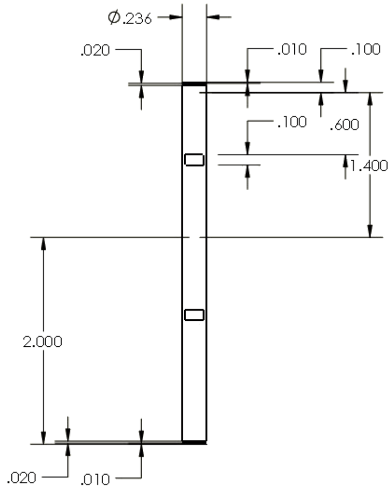


SOLIDWORKS Steps:

Create a rectangle the correct outside dimensions
Extrude boss it out the correct depth
Extrude cut the holes

Fabrication Steps: None: In order to speed up production time all of our components mounting holes were one inch apart. This allowed us to reuse an existing 6 x 9 inch plate of 1/18" thick aluminum with a grid pattern of holes already drilled in it 1" on center. This saved time and allowed us to easily make changes as needed.

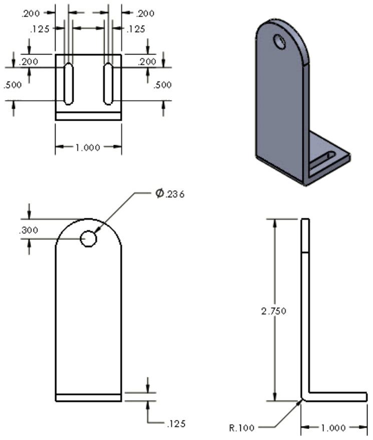
Claw System Axle



SOLIDWORKS Steps:
Extrude boss the length of the axle
Extrude cut the flats with an offset plane
Revolve cut the C-Clip slots
Chamfer the ends

Fabrication Steps:
Cut stock slightly oversize
Turn down to length
Chamfer ends
Mount in mill vice with just the end sticking out
Mill flats

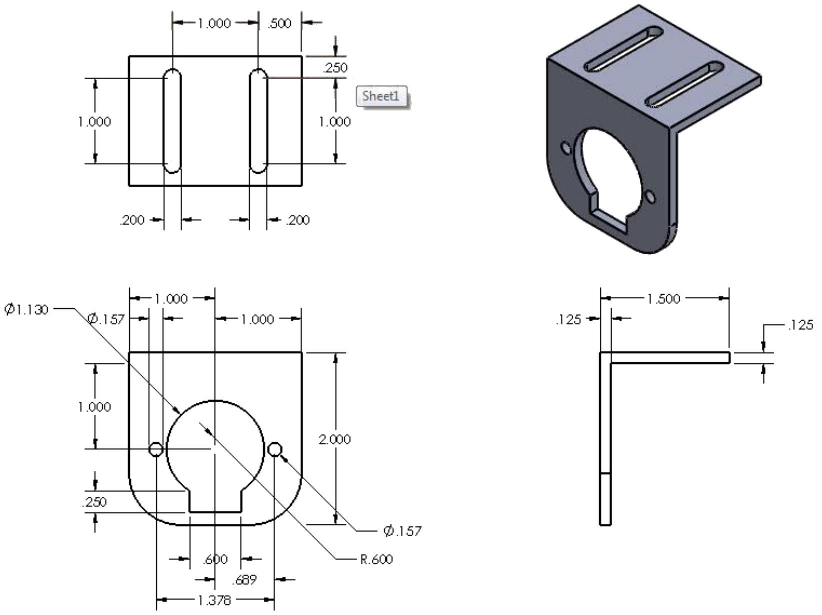
Claw System Support Brackets



SOLIDWORKS Steps:
Thin Extrude the body
Extrude cut the holes
Fillet the body corners
Extrude cut slots

Fabrication Steps:
Cut 1" square stock slightly oversize
Mill out material
Drill the axle hole (#29 drill)
Mill slots

Rear Stepper Motor Bracket



SOLIDWORKS Steps:

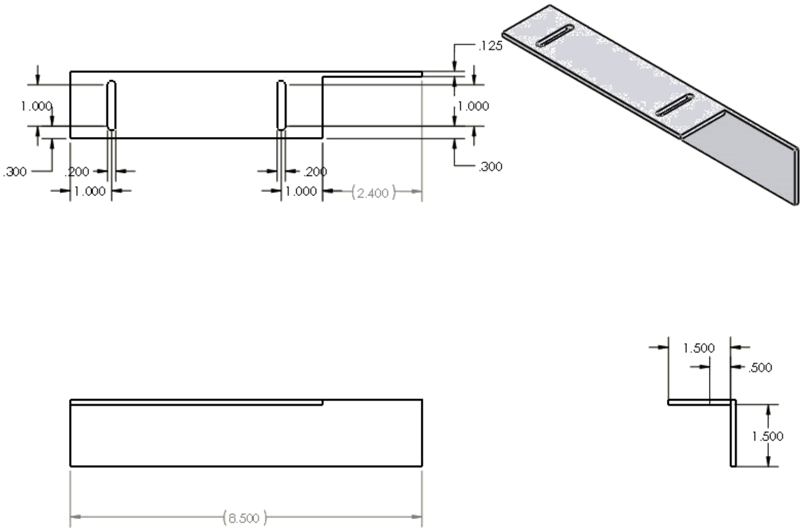
Thin Extrude the body
Extrude cut the holes
Fillet the body corners
Extrude cut the slots.

Fabrication Steps:
Load SOLIDWORKS file into CAMWorks
Automatic feature recognition
Generate tool paths
Post G-Code
Upload to TorMac
Monitor machine.

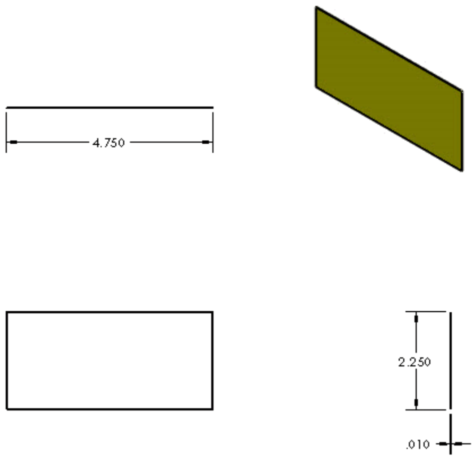
Rear Bucket Slider Bracket

SOILDWORKS Steps:
Thin Extrude the body
Extrude cut the holes
Fillet the body corners

Fabrication Steps:
Cut stock slightly oversize
Mill to size
Center drill all holes
Drill all holes (#29 Drill)
Sand fillets



Rear Bucket Slider Bracket



SOLIDWORKS Steps:

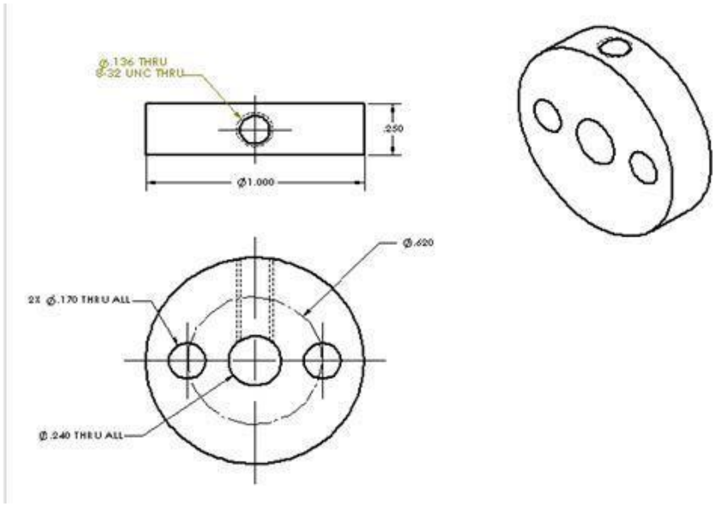
Create a rectangle the correct outside dimensions

Extrude boss it out the correct depth.

Fabrication Steps:

Mark dimensions on sheet metal Cut to specs with tin snips.

Front Wheel Hub



SOLIDWORKS Steps:

Boss Extrude the body Cut Extrude the holes.

Fabrication Steps:

Face the current side

Turn to correct diameter

Center drill axle hole

Drill axle hole (B drill)

Plunge cut two slightly oversized hubs off of stock

Face the opposite side and bring to correct thickness

Mount with rear wheel in jig Drill two holes on either side of the axle hole (#18 drill)

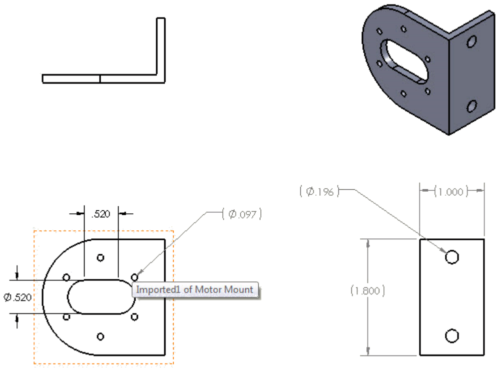
Mount in mill vice

Center drill and drill pilot hole (#29 drill)

Thread hole (8-32 UNC tap).

*We did not fabricate these parts, but rather re-used them from an old ENR-153 robot in order to save time and production cost.

Front Wheels



SOLIDWORKS Steps:

Boss Extrude the body

Cut Extrude the holes

Hole Wizard the threaded holes

Chamfer the edges.

Fabrication Steps:

Face the current side

Center drill axle hole

Drill axle hole (15/64 drill)

Face opposite side and bring to correct thickness

Mount both front wheels in fixture

Turn down outside diameter and chamfer edges

Mount in jig with hubs

Center drill and drill two pilot holes (#29 drill)

Remove from jig and thread holes (8-32 UNC tap).

*We did not fabricate these parts, but rather re-used them from an old ENR-153 robot in order to save time and production cost.

Drive Motor Brackets

SOLIDWORKS Steps:

Thin Extrude the body

Extrude cut the holes

Fillet the body corners

Extrude cut the slots

Fabrication Steps:

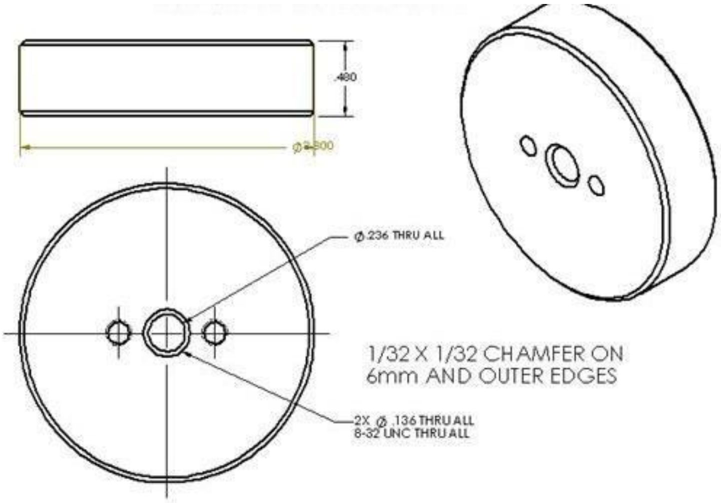
Load SOLIDWORKS file into CAMWorks

Automatic feature recognition

Generate tool paths

Post G-Code Upload to TorMac

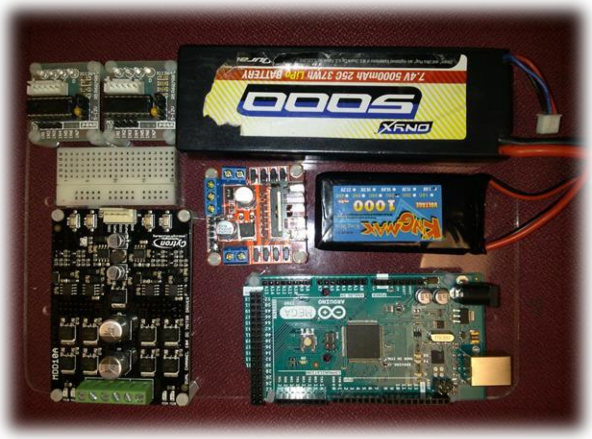
Monitor machine.



Electronics Mount Plate

Fabrication:

All electronics mocked up on 1/18" Lexan
All holes marked
All holes drilled with #18 drill
Electronic mounted using plastic standoffs



For the following components and hardware we opted to use off the shelf components from local hardware stores as well as online retailers.

Claw Stepper Support Brackets

SOLIDWORKS Steps:

The following component was not modeled in our SOLIDWORKS model as the motor that was initial going to be used did not have the strength or precision we required. This was determined after testing the fully designed robot and off the shelf components were used.

Fabrication Steps:
No fabrication was necessary as the L-Brackets were purchased from HomeDepot.

Bucket System Drawer Slides

SOLIDWORKS Steps:

We used an existing model off of GrabCad.com in order to reduce drafting time. The model was resized using the scale feature in SOLIDWORKS to fit our required dimensions.

Fabrication Steps:

No fabrication was necessary as the Drawer slides we purchased through amazon.com.

Hardware

All bolts, nuts, and washers were purchased from HomeDepot. The following size hardware was used to assemble the robot:

Bolts:

6-32 x ½ Inch UNC
8-32 x 3 Inch UNC
10-32 x 1.5 Inch UNC

Nuts:

6-32 UNC
8-32 UNC
10-32 UNC

Washers:

#6 flat washers
#6 locking washers
#8 flat washers
#8 locking washers
#10 flat washers
#10 locking washers

DESIGN ANALYSIS

Table 1:

This table describes Robot Testing 4, the only official testing we participated in. We did not participate in earlier testing modules because our goal was to build a robot for ASEE and not TYESA which led to more intricate design procedures.

Overall Scores

Team #	Instructor	Team Name	Exhibit Score	Testing Score	Total Points	Team Rank
4	Kumar	Bag n' Tag	0.00	95.00	95.00	1
8	Gamory	Reel Time	0.00	65.00	65.00	2
2	Gamory	Deadliest Catch	0.00	15.00	15.00	3
1	Wadach	Fishy Engineering	0.00	5.00	5.00	4
3	Kumar	Murdoch's Minions	0.00	0.00	0.00	5
5	Wadach	Megaladon	0.00	0.00	0.00	5
6	Wadach	HMS Victory	0.00	0.00	0.00	5
7	Kumar	The Original Deadliest Catch™	0.00	0.00	0.00	5
9	Gamory	To Be Announced	0.00	0.00	0.00	5
10	Gamory	The Professional Perfectionists	0.00	0.00	0.00	5
11	Gamory	Starfish	0.00	0.00	0.00	5
12	Gamory	PK4	0.00	0.00	0.00	5
13	Gamory	The A Squad	0.00	0.00	0.00	5
14	Gamory	Shark Week	0.00	0.00	0.00	5
15	Kumar	Specifically Nick's Team	0.00	0.00	0.00	5
16	Wadach	Fisherman's Friend	0.00	0.00	0.00	5
17	Flosenzier	Night Shift	0.00	0.00	0.00	5
18	Flosenzier	Sad Chuckle	0.00	0.00	0.00	5

Although we were not able to rank very well, our robot showed promise as the only robot who had a color sorting mechanism present on the bot. We are currently going through great amounts of refining to ensure that the robot performs in the near future.

Table 2:

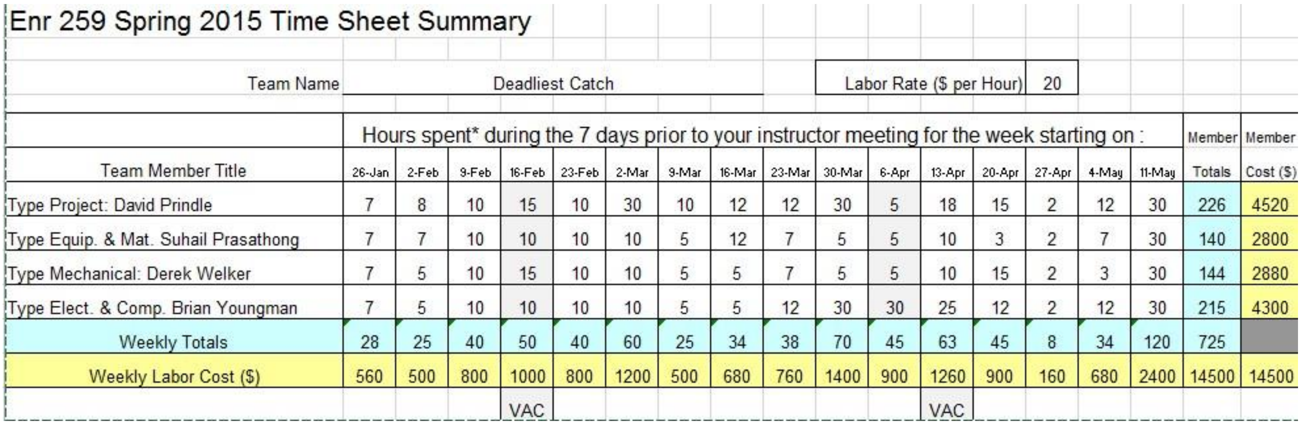
This table contains an intricate bill of materials that includes part name, supplier, part number or size, unit cost and total cost. It encompasses everything utilized to complete the project.

Name	Quantity/Model	Supplier	Cost
Bucket	1	CADimensions	Sponsored
Wheels	1	CADimensions	Sponsored
Vibration Motor Bracket	1	CADimensions	Sponsored
Claw	1	CADimensions	Sponsored
Slide Tray	1	CADimensions	Sponsored
Rear axle assembly	1	CADimensions	Sponsored
Aluminum Sheet Plate 1/8" (.125) X 9" X 12" 9X12	2	Amazon.com	$19.00
60863 1-1/2-Inch by 1-1/2-Inch by 1/16-Inch by 48-Inch Angle	1	Amazon.com	$18.22
6061 Aluminum Round Bar, 1/4" Diameter, 36" Length	1	Amazon.com	$10.50
1" x 1", 6061 T6511 Extruded Aluminum Square Bar	1	Amazon.com	$10.00
2 Pcs 10" 3-fold Full Extension Ball Bearing Drawer Slides	1	Amazon.com	$8.11
3 in. Zinc-Plated Corner Brace (4-Pack)	1	HomeDepot	$3.77
#8-32 x 3 in. Phillips-Slotted Round-Head Machine Screws	6	HomeDepot	$1.96
#8-32 tpi Zinc-Plated Machine Screw Nut (100-Piece)	1	HomeDepot	$3.93
#8 Zinc-Plated Steel Flat Washer (30-Pack)	1	HomeDepot	$0.98
#6-32 x 1/2 in. Phillips Flat-Head Machine Screws (370-Pack)	1	HomeDepot	$6.76

#10-32 Zinc Plated Machine Screw Nut (100-Pieces)	1	HomeDepot	$4.72
#10-32 x 1-1/4 in. Phillips Flat-Head Machine Screws (100 Pieces)	1	HomeDepot	$4.92
Cytron 10A Dual Motor conroller	MDD10A	Amazon.com	$27
DC Gear Head 40:1 6v motor	KIT-MTR-36-06-180	TrossenRobotics.com	$29.95
Pixy Camera	CMUcam5	Amazon.com	$70.00
7.4V 1000mAh battery	PRT-10472	Robotshop.com	$10.00
7.4V 5000mAh battery	DTXC1864	Amazon.com	$44.00
DPDT Toggle Switch	20 Amp	Amazon.com	$1.00
Breadboard Jumper Wire	Variety	Robotshop.com	$5.00
5V DC Stepper Motor	28BYJ-48	Amazon.com	$3x2
5V 1A Stepper Motor	23A-6102T	Robotshop.com	$29.95
EasyDriver	80343	Amazon.com	$14.95
Arduino MEGA	MEGA	Amazon.com	$45.00
Futaba Servo	S3003	Robotshop.com	$11.00
Total Cost:	**NA**	**NA**	**$380.72**

Conclusively, we were able to keep our budget to a fairly reasonable reach being slightly over $380. Considering the amount of components and parts our robot has, it comes out to be a pretty effective product considering it functions successfully eventually. To reiterate, we have every intention to ensure that this robot runs successfully. **Table 3:**

This table contains the final labor cost analysis that portrays the number of hours, cost and effort put into the project relative to a real life engineering prospect.



Enr 259 Spring 2015 Time Sheet Summary

Team Name Deadliest Catch

Labor Rate ($ per Hour) 20

	Hours spent* during the 7 days prior to your instructor meeting for the week starting on :																Member	Member
Team Member Title	26-Jan	2-Feb	9-Feb	16-Feb	23-Feb	2-Mar	9-Mar	16-Mar	23-Mar	30-Mar	6-Apr	13-Apr	20-Apr	27-Apr	4-May	11-May	Totals	Cost ($)
Type Project: David Prindle	7	8	10	15	10	30	10	12	12	30	5	18	15	2	12	30	226	4520
Type Equip. & Mat. Suhail Prasathong	7	7	10	10	10	10	5	12	7	5	5	10	3	2	7	30	140	2800
Type Mechanical: Derek Welker	7	5	10	15	10	10	5	5	7	5	5	10	15	2	3	30	144	2880
Type Elect. & Comp. Brian Youngman	7	5	10	10	10	10	5	5	12	30	30	25	12	2	12	30	215	4300
Weekly Totals	28	25	40	50	40	60	25	34	38	70	45	63	45	8	34	120	725	
Weekly Labor Cost ($)	560	500	800	1000	800	1200	500	680	760	1400	900	1260	900	160	680	2400	14500	14500
				VAC								VAC						

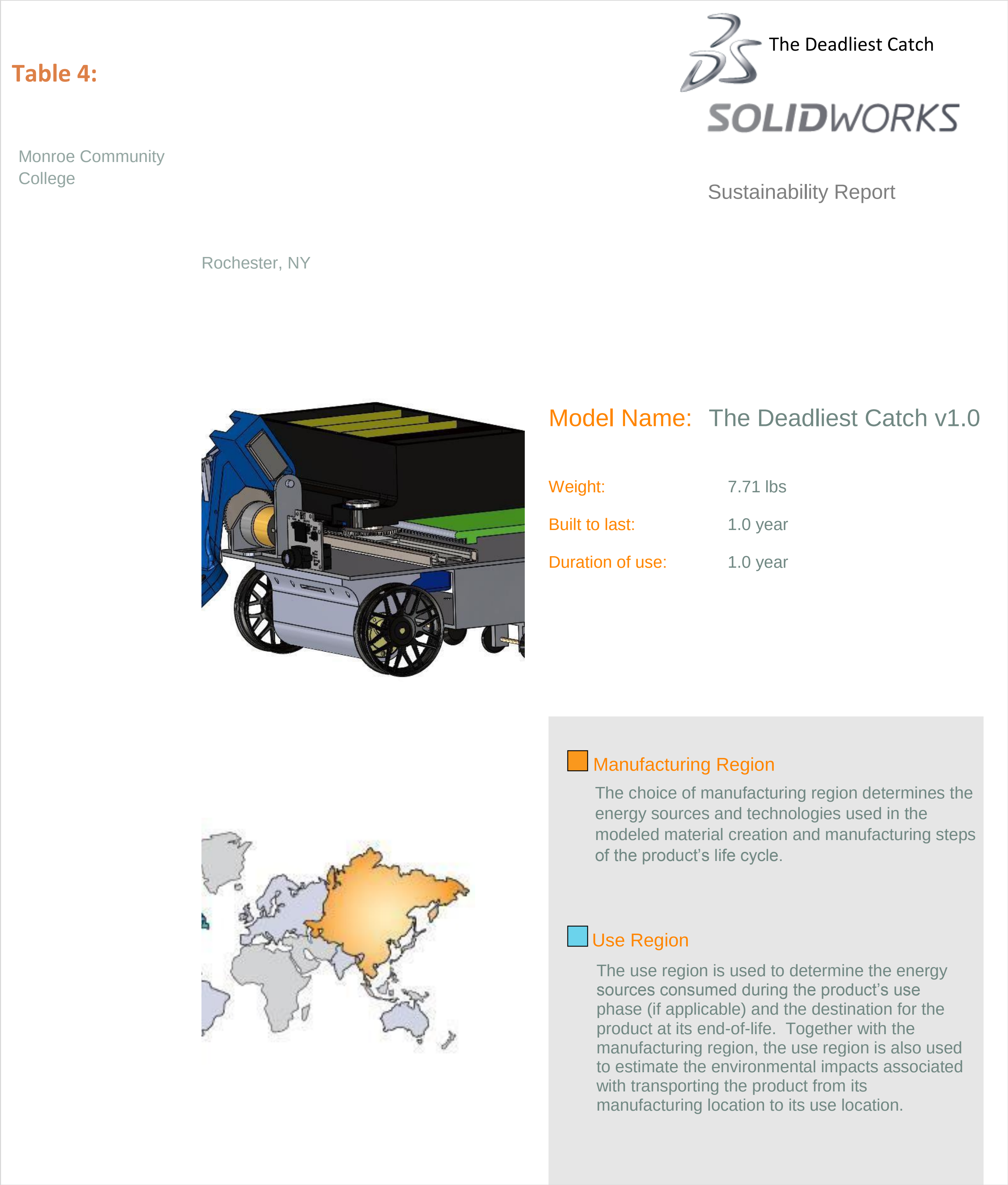


Table 4:

The Deadliest Catch

SOLIDWORKS

Monroe Community College

Sustainability Report

Rochester, NY

Model Name: The Deadliest Catch v1.0

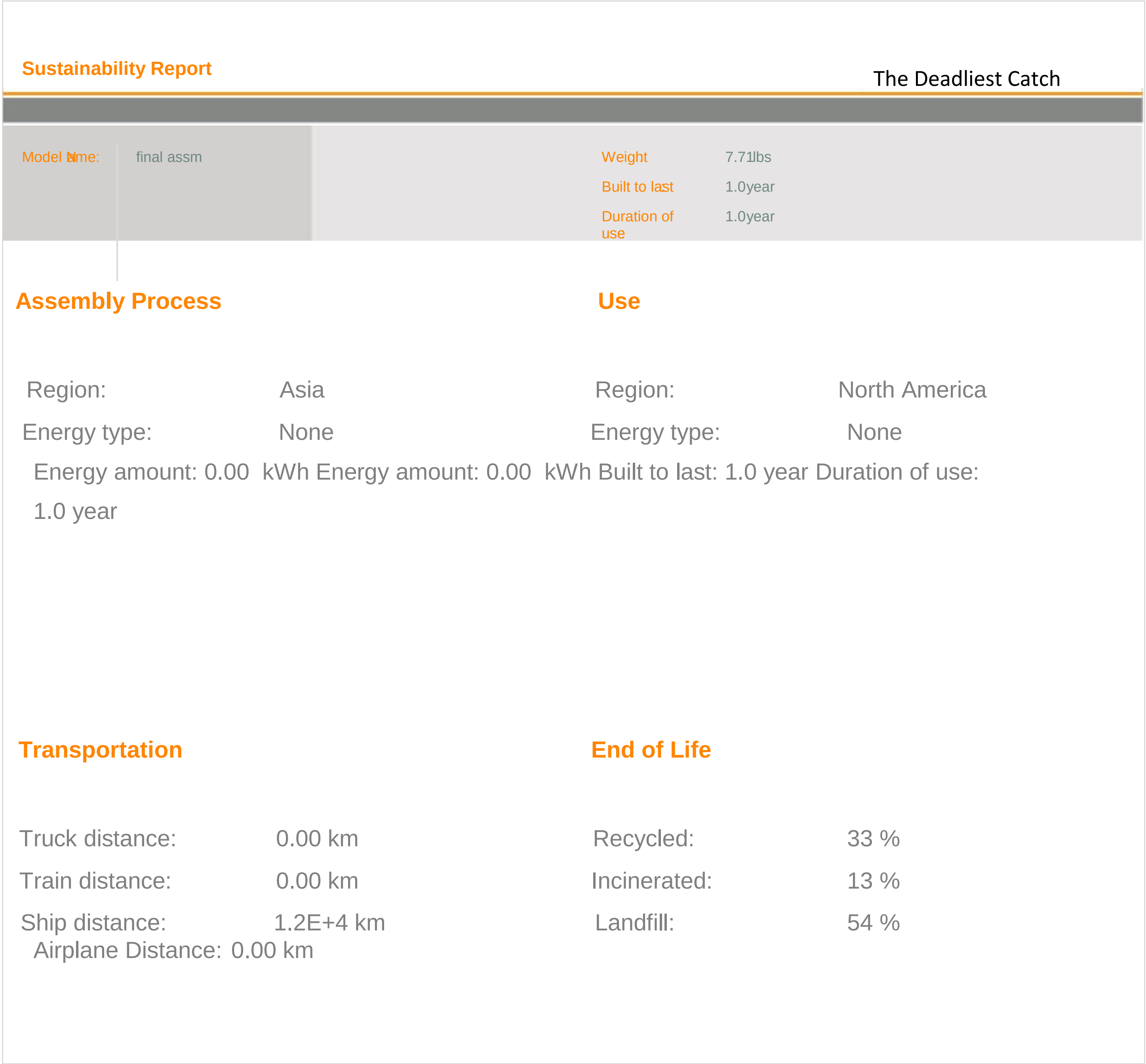
Weight:	7.71 lbs
Built to last:	1.0 year
Duration of use:	1.0 year

🟧 Manufacturing Region

The choice of manufacturing region determines the energy sources and technologies used in the modeled material creation and manufacturing steps of the product’s life cycle.

🟦 Use Region

The use region is used to determine the energy sources consumed during the product’s use phase (if applicable) and the destination for the product at its end-of-life. Together with the manufacturing region, the use region is also used to estimate the environmental impacts associated with transporting the product from its manufacturing location to its use location.



Sustainability Report

Model Name:	final assm	Weight	7.71lbs
		Built to last	1.0year
		Duration of use	1.0year

Assembly Process

Use

Region:	Asia	Region:	North America
Energy type:	None	Energy type:	None

Energy amount: 0.00 kWh Energy amount: 0.00 kWh Built to last: 1.0 year Duration of use: 1.0 year

Transportation

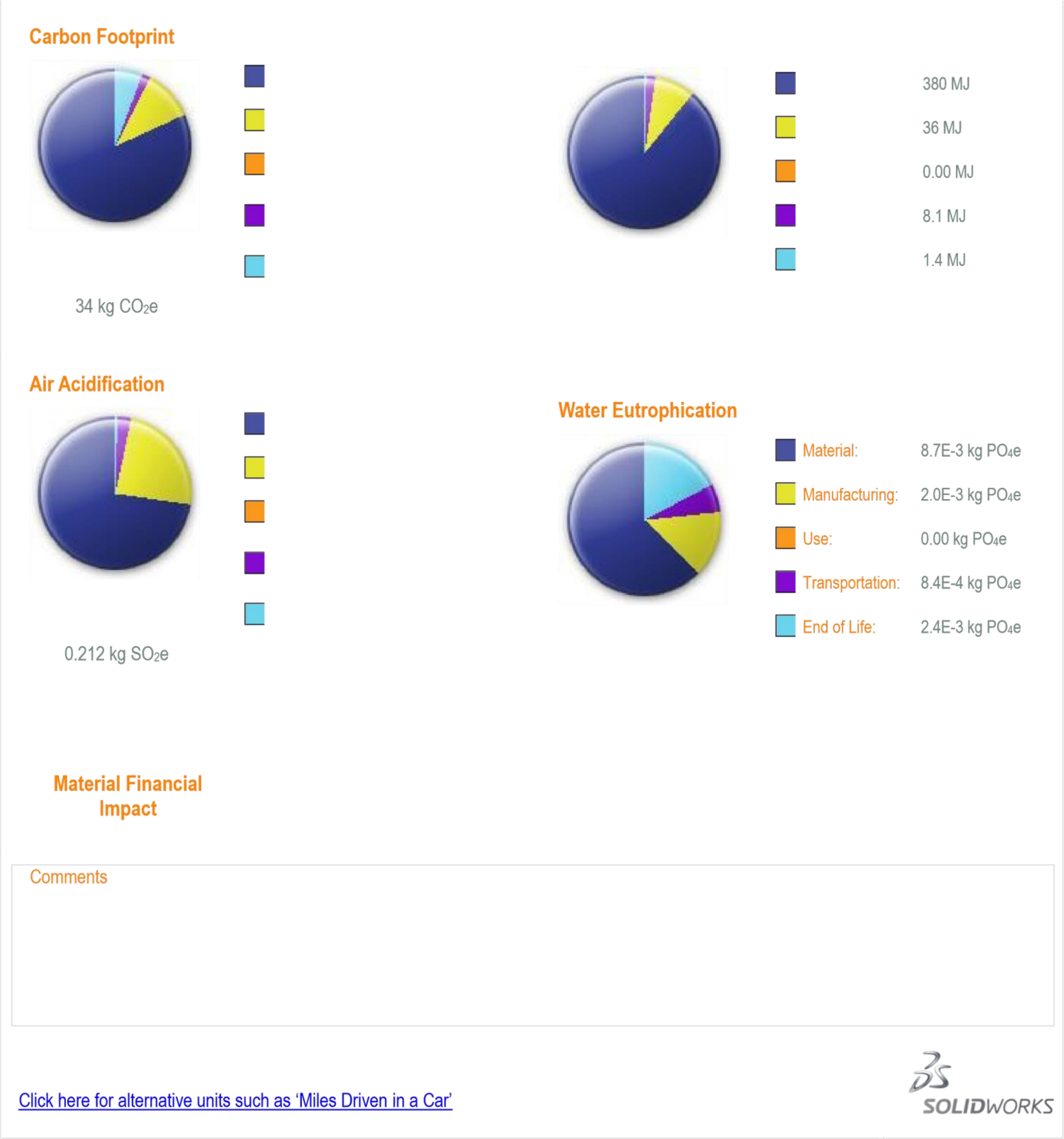
End of Life

Truck distance:	0.00 km	Recycled:	33 %
Train distance:	0.00 km	Incinerated:	13 %
Ship distance:	1.2E+4 km	Landfill:	54 %
Airplane Distance:	0.00 km		

Sustainability Report

Model Name:	final assm	Weight	7.71lbs
		Built to last	1.0year
		Duration of use	1.0year

Environmental Impact (calculated using CML impact assessment methodology)



Carbon Footprint

34 kg CO$_2$e

🟦	380 MJ
🟨	36 MJ
🟧	0.00 MJ
🟪	8.1 MJ
🟦	1.4 MJ

Air Acidification

0.212 kg SO$_2$e

Water Eutrophication

🟦	Material:	8.7E-3 kg PO$_4$e
🟨	Manufacturing:	2.0E-3 kg PO$_4$e
🟧	Use:	0.00 kg PO$_4$e
🟪	Transportation:	8.4E-4 kg PO$_4$e
🟦	End of Life:	2.4E-3 kg PO$_4$e

Material Financial Impact

Comments

Click here for alternative units such as 'Miles Driven in a Car'

Total Energy Consumed

Material: 28 kg CO$_2$e

Material:

Manufacturing: 3.6 kg CO$_2$e

Manufacturing:

Use: 0.00 kg CO_2e

Transportation: 0.606 kg CO_2e

End of Life: 1.9 kg CO_2e

Use:

Transportation:

End of Life:

420 MJ

Material: 0.154 kg SO_2e

Manufacturing: 0.051 kg SO_2e

Use: 0.00 kg SO_2e

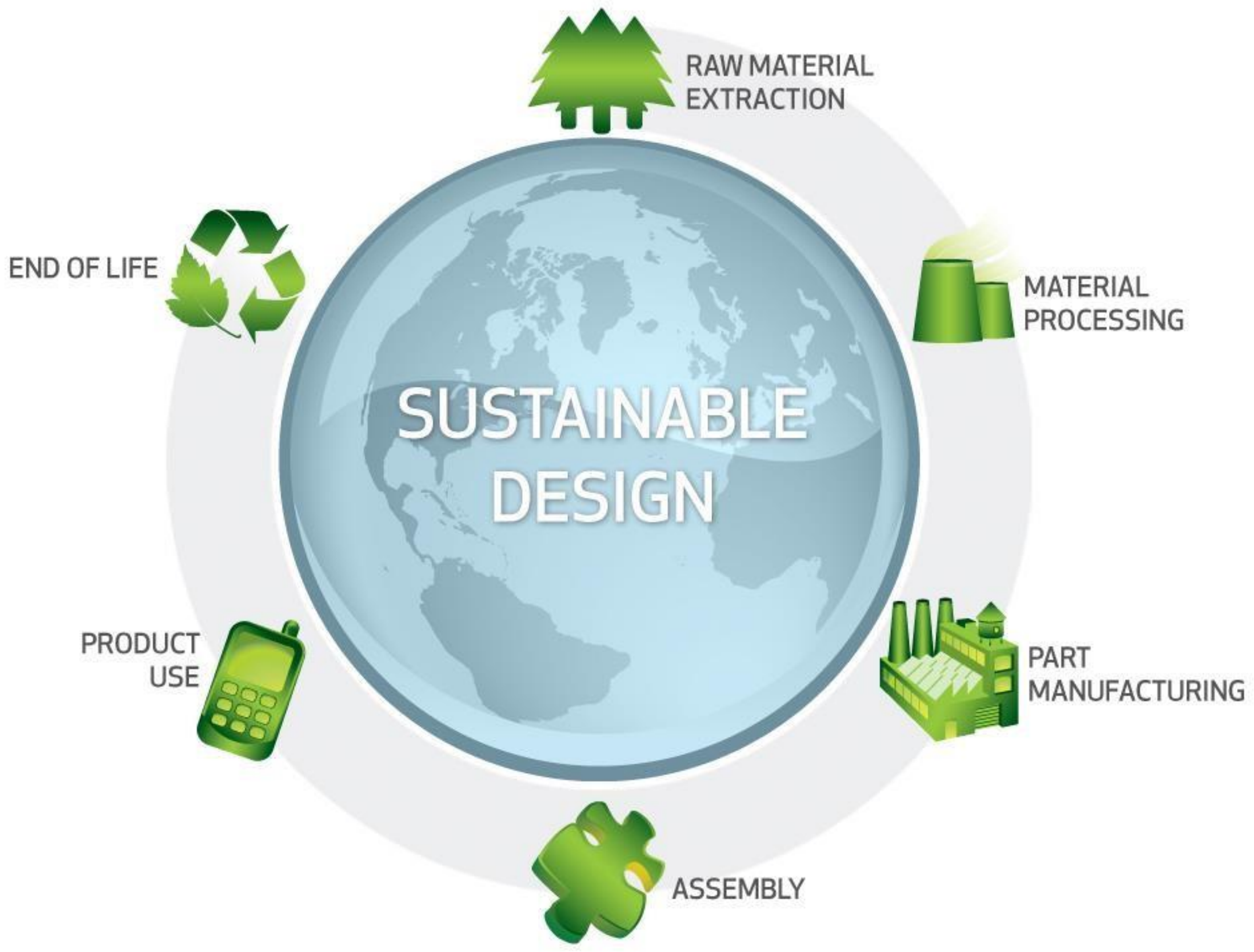
Transportation: 6.0E-3 kg SO_2e

End of Life: 9.8E-4 kg SO_2e

0.014 kg PO_4e

3.80 USD

Component	Carbon	Water	Air	Energy
chassis plate	12	2.7E-3	0.081	140
holder	4.9	2.7E-3	0.025	73
DC_Gear_Motor_6V_40-1	2.5	5.8E-4	0.018	31
slide	1.6	9.0E-4	8.0E-3	24
Slider Bracket	1.8	4.2E-4	0.013	22
Futaba S3004 Servo	1.1	2.6E-4		8.0E-3
Motor	14			
300mm Ball bearing	0.576	6.7	4.4E-4	3.3E-3
runner				
Stepper 28BYJ48	0.392	3.6E-4 4.6		2.2E-3
Claw Half Body bottom	0.441	2.3E-4	2.7E-3	7.6
linear stepper bracket	0.586	1.3E-4 4.1E-3 7.2		


RAW MATERIAL EXTRACTION
MATERIAL PROCESSING
PART MANUFACTURING
ASSEMBLY
PRODUCT USE
END OF LIFE
SUSTAINABLE DESIGN

Glossary

Air Acidification - Sulfur dioxide, nitrous oxides other acidic emissions to air cause an increase in the acidity of rainwater, which in turn acidifies lakes and soil. These acids can make the land and water toxic for plants and aquatic life. Acid rain can also slowly dissolve manmade building materials such as concrete. This impact is typically measured in units of either kg **sulfur dioxide equivalent (SO_2), or moles H+ equivalent**.

Carbon Footprint - Carbon-dioxide and other gasses which result from the burning of fossil fuels accumulate in the atmosphere which in turn increases the earth's average temperature. Carbon footprint acts as a proxy for the larger impact factor referred to as Global Warming Potential (GWP). Global warming is blamed for problems like loss of glaciers, extinction of species, and more extreme weather, among others.

Total Energy Consumed - A measure of the non-renewable energy sources associated with the part's lifecycle in units of megajoules (**MJ**). This impact includes not only the electricity or fuels used during the product's lifecycle, but also the upstream energy required to obtain and process these fuels, and the embodied energy of materials which would be released if burned. PED is expressed as the net calorific value of energy demand from non-renewable resources (e.g. petroleum, natural gas, etc.). Efficiencies in energy conversion (e.g. power, heat, steam, etc.) are taken into account.

Water Eutrophication - When an over abundance of nutrients are added to a water ecosystem, eutrophication occurs. Nitrogen and phosphorous from waste water and agricultural fertilizers causes an overabundance of algae to bloom, which then depletes the water of oxygen and results in the death of both plant and animal life. This impact is typically measured in either kg **phosphate equivalent (PO_4) or kg nitrogen (N) equivalent**.

Life Cycle Assessment (LCA) - This is a method to quantitatively assess the environmental impact of a product throughout its entire lifecycle, from the procurement of the raw materials, through the production, distribution, use, disposal and recycling of that product.

Material Financial Impact - This is the financial impact associated with the material only. The mass of the model is multiplied by the financial impact unit (units of currency/units of mass) to calculate the financial impact (in units of currency).

Learn more about Life Cycle Assessment

This table contains the environmental impact for all the fabricated parts on our robot. We utilized the SolidWorks Sustainability tools to gather data for our Carbon Footprint, Energy Usage, Air Acidification and Water Eutrophication for all the fabricated parts.

Conclusively, we feel that our design process and analysis was pretty thorough and complete. We followed all the right processes and procedures and all we have left to do is successfully implement everything that has been in development for months.

TEAM AND TECHNICAL ANALYSIS

Team Functionality

Overall, we believe that our team functioned pretty decently. We had two mechanical engineers and two computer engineers. Our team dynamic was that we broke into sub-teams. The mechanical and the electrical/computer teams. The mechanical team was responsible for building the robot based off the electrical components we wanted and the software perspective we wanted to take.

As far as the mechanical sub-team is concerned, the work was pretty much divided up equally and there was fluid communication between the members. The electrical and software team was led by Brian Youngman, while Suhail Prasathong served as his helper fulfilling tasks assigned by Youngman. We found this approach to be effective because it is hard for two people to work on the same block of code due to different coding styles, logic and understanding of the programming systems. In most cases, two people working on the same code results in disagreement and chaos. Therefore, we decided that having two different entities with one being superior will ensure that the other is not deadweight on the team and that the entire affair is productive for all parties involved.

Generally, we feel that our team worked well. Obviously, throughout the span of the project we ran into a few problems in communication and we occasionally had a few misunderstandings but as grown mature engineers we were able to solve out these differences through intensive meetings that eventually led to desirable conclusions.

Problems & Corrections

Like any other group effort, we ran into various problems both technical and behavioral. One major problem we ran into was the PIXY camera. We found that the product did not perform as advertised, we were especially shocked due to the fact that there were various reviews online confirming that the product did what it was supposed to do. However, upon use, we found that the information was not true. As we dug further, we found that there were many people suggesting that all those comments made were people who used the Raspberry Pi as opposed to the Arduino microcontroller. The Raspberry Pi microcontroller had more libraries and accessibility for the PIXY camera than the Arduino did and this was something we were not aware about. We resolved this problem by developing a quick fix temporary solution for the camera and decided to eventually switch over the Raspberry Pi after Robot Testing 4 was complete.

Another major problem our robot faced was the size of the robot. We were under the impression that we can place the bot into the box and then remove it for the run. However, that was simply untrue and our robot is going through a revision in mechanical design where we will be cutting off two inches on the sides so the bot can be accommodated without much issue.

One last overwhelming concern we found was with communication amongst the team. Although it worked out great early on and our team was pretty cohesive throughout the first half of the semester, as soon as the breaks started to come in, we found that there were unwanted gaps of communication. We resolved this issue by simply establishing more virtual communication through emails and texts which helped bring the group's agenda back on track.

Overall, although we had various problems, we felt that as a group we were able to get through the issues with the determination that we had to try to make the robot work eventually. For the most part, although the robot is still a work in progress, we were able to make good headway on it.

Changes & Methodology

From a very honest perspective, we do not believe we would change the team dynamic in any way. We truly believe that although we do not have a final working product, we have established a tremendous working relationship that is bound to take us a long way. We have confidence in the team's understanding and hope to complete the robot.

Although we have implemented the right components and methods to solve the overall problem, there have still been situations where we have run into various problems and errors. Our camera, although was a good component, the implementation could have been better. We should have done further research on its problems with the Arduino board and we should've just used the Raspberry Pi from the get go. Nevertheless, we do plan on building upon this.

Design Decisions

From a very honest perspective, we do not believe we would change the team dynamic in any way have affected the creation of the robot. Every design decision was successful except the implementation of the PIXY and the dimensions of the robot. Those two decisions were the only two major decisions that did not quite go our way. Once we recover from those detrimental mistakes, we will be able to further develop our robot and come out with a conclusive solution.

Project Re-Implementation

If we were given another opportunity to re-approach the robot, we would first ensure to fit within the dimensions and we will definitely ensure to start with the Raspberry Pi. At this point, our robot still has potential and is under development. We have clearly been set back a while due to the mistakes we have made early on. However, with determination and time put into the current model, we should have a working bot before the American Society for Engineering Education's National Robotics Competition.