UNIVERSITY OF
WATERLOO

Department of Mechanical
and Mechatronics Engineering

Fishing Robot

A Report Prepared For:
MTE 100/121 Final Project

Report Prepared By:

Group 8-10

Aryaan Ray 21224351

Ian Martin 21127706

Liam Doyle 21153052

Rahul Patel 21118445

Contents

List of Figures

Summary

The robot that the team constructed is an automated fishing robot prototype. The robot was made from LEGO and powered by a LEGO EV3 controller and motors. The report contains the organization and flow of the project, technical details of the robot, a breakdown of the software, and changes and recommendations the team has made. The

robot was a fully functional prototype but was limited by the inherent mechanical disadvantages of LEGO.

1.0 Introduction

The purpose of this project is to automate the task of fishing via a mechatronics system to help increase the number of fish an angler can catch. The basic movements of fishing, such as casting, waiting for feedback from a fish, and subsequently reeling the fish back, could be automated using motors and sensors. Therefore, the team decided to automate fishing, both because of the uniqueness of creating a robot that can fish, and its ability to use a variety of sensors needed to automate this task.

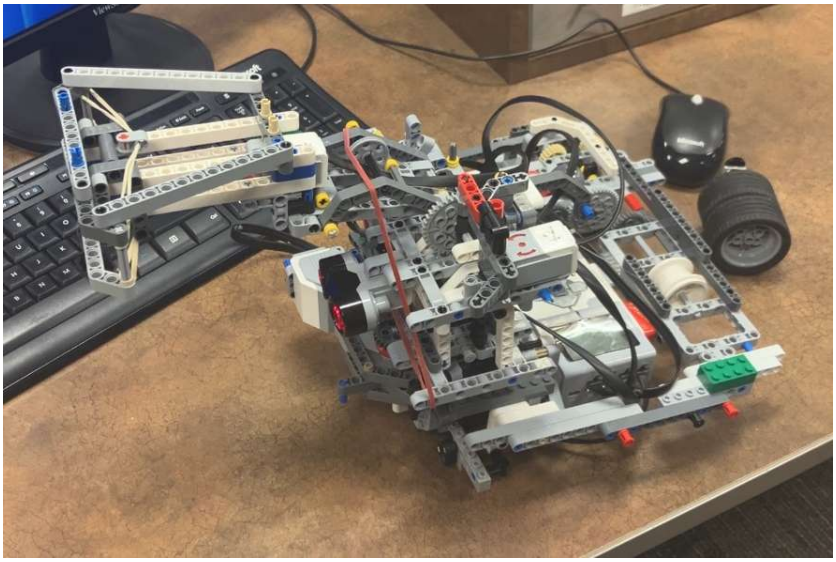

Figure 1: Robot Isometric View

2.0 Scope

2.1 Functionality

Fishing with a rod and a reel includes the following tasks: casting, reeling, vertical adjusting, horizontal (lateral) adjusting, and a way to reset the process. Therefore, the fishing robot must be able to perform similar tasks. The robot can cast, reel, rotate upon a turret, raise the arm, and reset the line and hook. An important subsystem of the robot is the cast clutch mechanism, which disengages the reeling system temporarily to cast the line with ease.

2.2 Inputs, Measurements, & Detections

The sensors that the robot uses to operate are as follows: a gyro sensor for measuring turret rotation, one motor encoder for measuring arm elevation, one motor encoder for detecting fish, an ultrasonic sensor for detecting objects in the way of the cast, a touch

sensor for detecting when the cast carriage is maximally pulled back, and the buttons on the EV3 for navigating the menu and operating the robot in manual mode. In manual mode, the buttons can be used to rotate the turret, raise, and lower the arm, and perform the cast and reel functions. To test the ultrasonic safety system, human input is required. Placing one's hand in front of the sensor and making sure it does not cast is the way the team ensured the system was working. Human input is also required to test the actuating of the reel function. The reel motor encoder must be rotated, simulating the pull of a fish on the line, to begin the reeling process. This process is done by hand.

2.3 Interactions

The robot uses four motors. One large motor is used for reeling, one large motor is used for rotating the turret, one large motor is used for elevating the arm, and one medium motor is used for engaging and disengaging the cast clutch. To cast, the robot engages the cast clutch and pulls back the cast carriage. The cast clutch is then disengaged, and the carriage is allowed to move freely, pulled by the tension of the elastic bands. This casts the line and hook. The turret motor rotates the entire assembly, allowing for latitudinal control, and the arm elevation motor controls the pitch of the arm, allowing for the "set the hook" motion, a fishing technique whereby the rod is rapidly moved upwards. The turret motor and arm elevation motor allow for full manual control. The reel motor encoder is programmed to trigger the reel function when it is back driven, simulating a fish pulling on the line. In practice, the motor's resistance is too difficult to overcome with the line, so manually turning the motor is a testable method of actuating the reeling system.

2.4 Completing Tasks & Shutdown Procedure

The first task the robot must complete is the startup procedure. When the robot is powered on, it first asks the user if the robot is in the proper startup position, which is the turret perpendicular to the EV3 brick and the arm parallel to the ground. If it is not, the robot gives the user temporary manual control to set this position. After this it prompts the user whether they want to fish in automatic mode or in manual mode, for which the user responds using the buttons on the EV3. If manual mode is selected, the user may set a position within predefined limits for the robot to cast from before pressing the centre button to enable automatic fishing from this position. If automatic mode is selected the robot proceeds into automatic fishing from the starting position. The robot casts the line and uses a timer to hold the line in the water for 30 seconds. It then reels it back and casts it again. Simultaneously, a second timer is tracking the overall fishing time, and when the timer reaches one hundred seconds, the shutdown procedure begins. Additionally, if a fish is caught, triggered by the reel motor encoder rotating, the shutdown procedure is called. The shutdown procedure consists of the following tasks. It returns the turret and arm

elevation motor back to the starting position determined in the startup procedure, actuates the reel motor, and slowly eases the tension out of the elastic bands, displays the amount of time the robot has been fishing for, and shuts down.

2.5 Changes in Scope

One of the most significant changes in scope that had to be performed was the scrapping of the lateral adjustment system. Originally, the team planned to have a feature where if the fish made lateral movements requiring the turret to rotate, the system would detect the motion of the fish and apply a countering effect using the turret motor. However, in practice, the way the turret was set up, with the LEGO turret inherently unstable, a high gear ratio to overcome, and a very resistive motor, the turret was nearly impossible to be moved by the fishing line and would have certainly broken apart with the amount of tension force needed. Therefore, the entire feature had to be scrapped, as there would have been no way to test and use it. The turret motor still proved to be useful for manual fishing and setting an ideal position for automatic fishing.

3.0 Constraints and Criteria

The robot constraints were modified throughout the entire construction process as new mechanical issues arose when integrating the different subsystems.

Originally, since the robot was not allowed to be tested on water due to safety concerns, the team initially abstracted the idea of the robot automatically fishing to the robot catching magnetic fish within 20 centimetres of its base.

The initial list of constraints are as follows:

1. The robot will have operational soft limits to prevent robot from breaking)
2. The robot should be able to self reset (calibration, casting, reeling, and looping this action for a set period before shutting down)
3. The turret must automatically zero its position
4. The robot must be able to place the fish in a basket.

The method of testing the ability of the robot to fish was initially modified to having the robot detect magnetic fish, which would be moved by a non-member of the group. However, due to the mechanical losses from the mechanical design of the robot (more information on section 4).

Self resetting counts only for the reeling, casting, and resetting position during shutdown. not automatic calibration on startup. This was due to the inability for the robot to

automatically calibrate without removing ultrasonic functionality. More information on this problem can be found in section 5.

Hence, this constraint was met by the implementation of the cast, reel, and shutdown function which automatically casts and resets the reel, as well as the mechanical design of the crossbow system allowing the robot to automatically charge its launch and release.

4.0 Mechanical Design and Implementation

4.1 Overview

The overall design methodology behind creating the robot was to replicate the simple configuration of a standard fishing rod with appropriate modifications to better suit it being made of LEGO and being automated with motors and a controller. At the basic level, the robot consists of the following subsystems: a casting mechanism, a reeling mechanism,

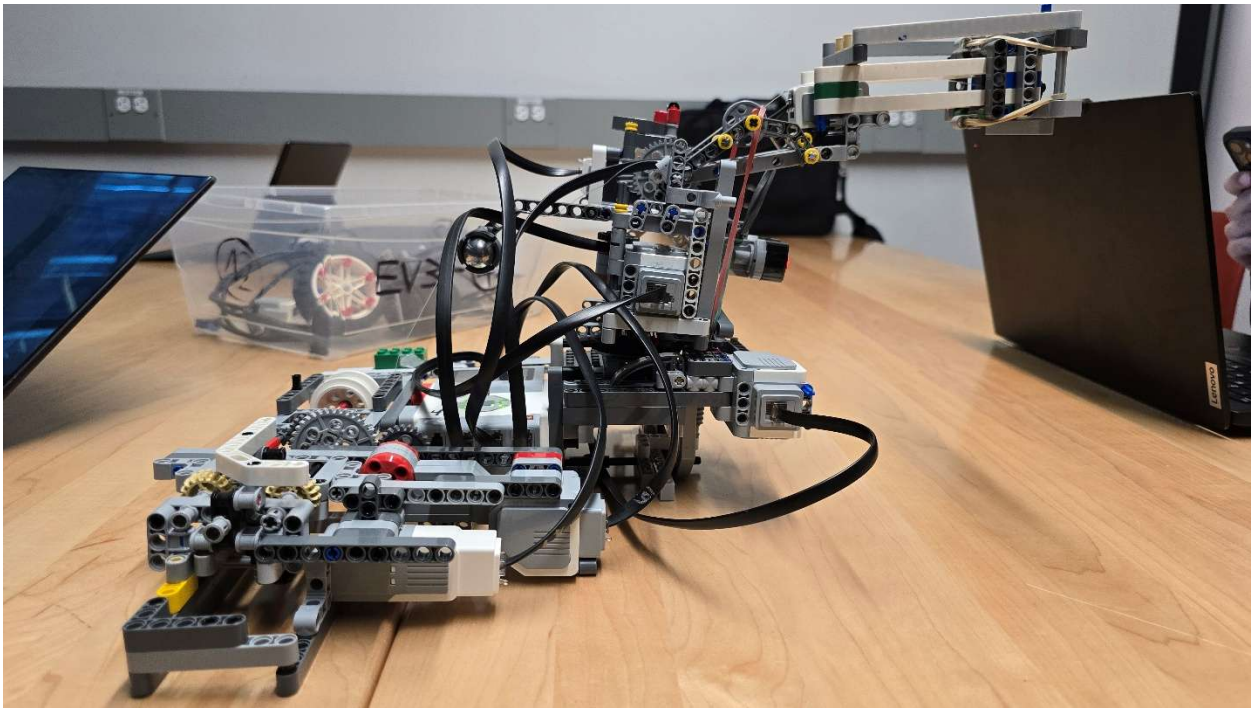

Figure 2: Robot Final Configuration

including the reel and the gearbox, an arm elevation mechanism, and a rotating turret mechanism. The turret system was mounted upon a LEGO base, and the casting and arm elevation system was built upon the rotating platform. The EV3 controller acted as a frame member and joined the arm assembly with the reel and the gearbox.

4.2 Casting Mechanism

The idea of a long, flexible arm that uses the tip velocity and mass of the lure to cast, present in a fishing rod, was explored but deemed too difficult to make from LEGO. The team settled on a “crossbow” style design, wherein the fishing line would pull in the lure which is seated on a carriage. The carriage is strung from the frame using elastic bands. As the line was reeled back, the carriage would store potential energy in the elastic bands. Once the line was disengaged, the carriage would accelerate forward, casting the lure out. The first method for sensing the carriage position was using a motor encoder to determine how far the carriage was pulled back using encoder counts. However, this system proved to

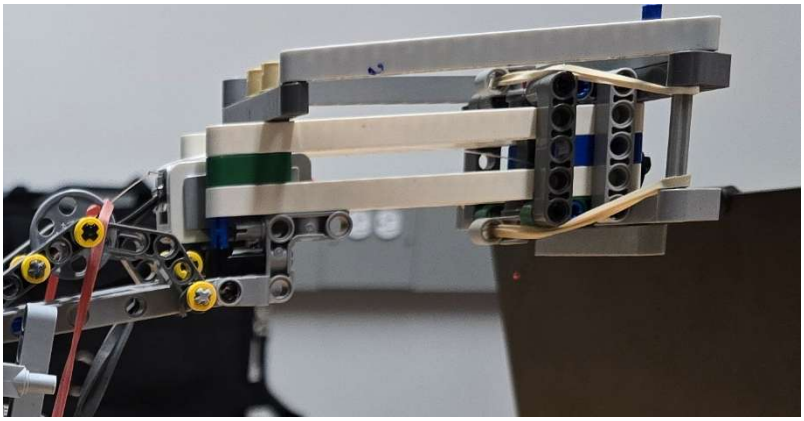

Figure 3: Casting Mechanism

be unreliable because there was slippage in the geartrain and every time the gearbox was disengaged, the encoder count would have to be reset. To solve this issue, a touch sensor was placed at the very back of the carriage’s travel, acting as a limit switch to determine when the carriage had reached maximum tension and could be released. As the carriage accelerates forward, the line is pulled from a reel. The casting mechanism was mounted at the end of the arm and used a wide frame, like a crossbow. The elastic bands were attached at one point at each end of the crossbow and one end on the carriage. The carriage was constrained to slide on a single-axis rail. The travel of the carriage was approximately eight centimetres.

4.3 Reel Gearbox/Cast Clutch

Upon building the casting mechanism, the team decided that the resistance from the reeling motor (LEGO large motor) was too much for the elastic bands to overcome when accelerating the carriage forward. The solution was to develop a system that could engage

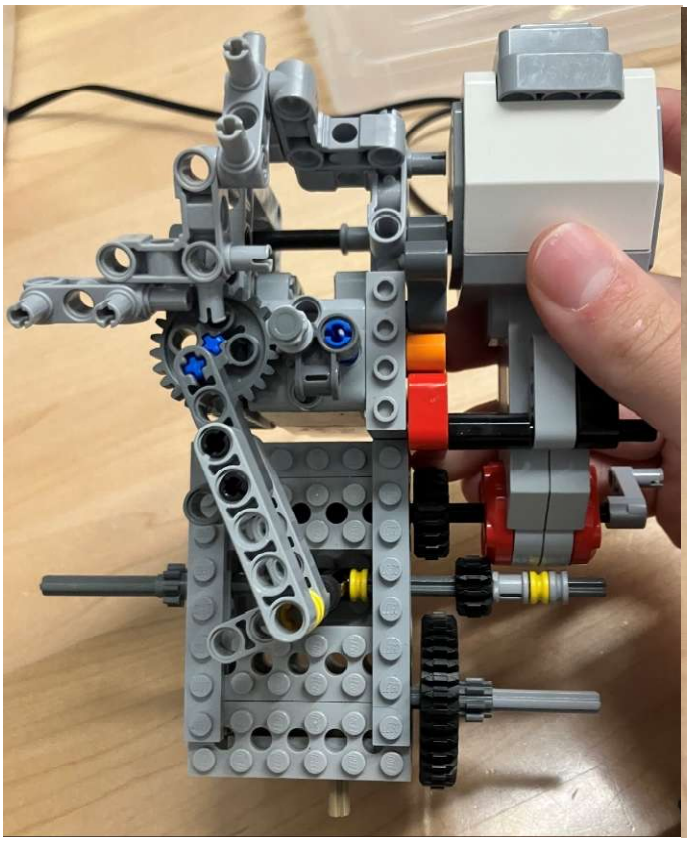

Figure 4: Gearbox First Iteration

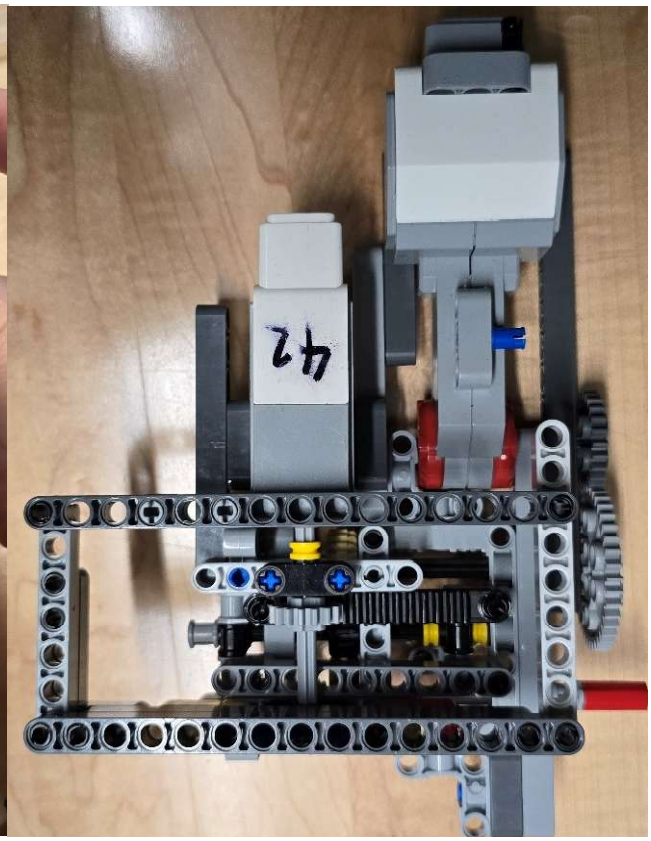

Figure 5: Gearbox Second Iteration

and disengage the reel from the reeling motor.

The system would need to allow the reel to freely rotate as the line was being casted, and then re-engage when the system needs to reel in the line. The mechanism that the team decided on was a clutch. The working principle behind the clutch system was a gear which would axially slide, meshing the driving gear to the driven gear. The first iteration (Figure 1) consisted of a medium LEGO motor actuating a linkage, which slid an idler gear in and out of mesh with the driving and driven gears.

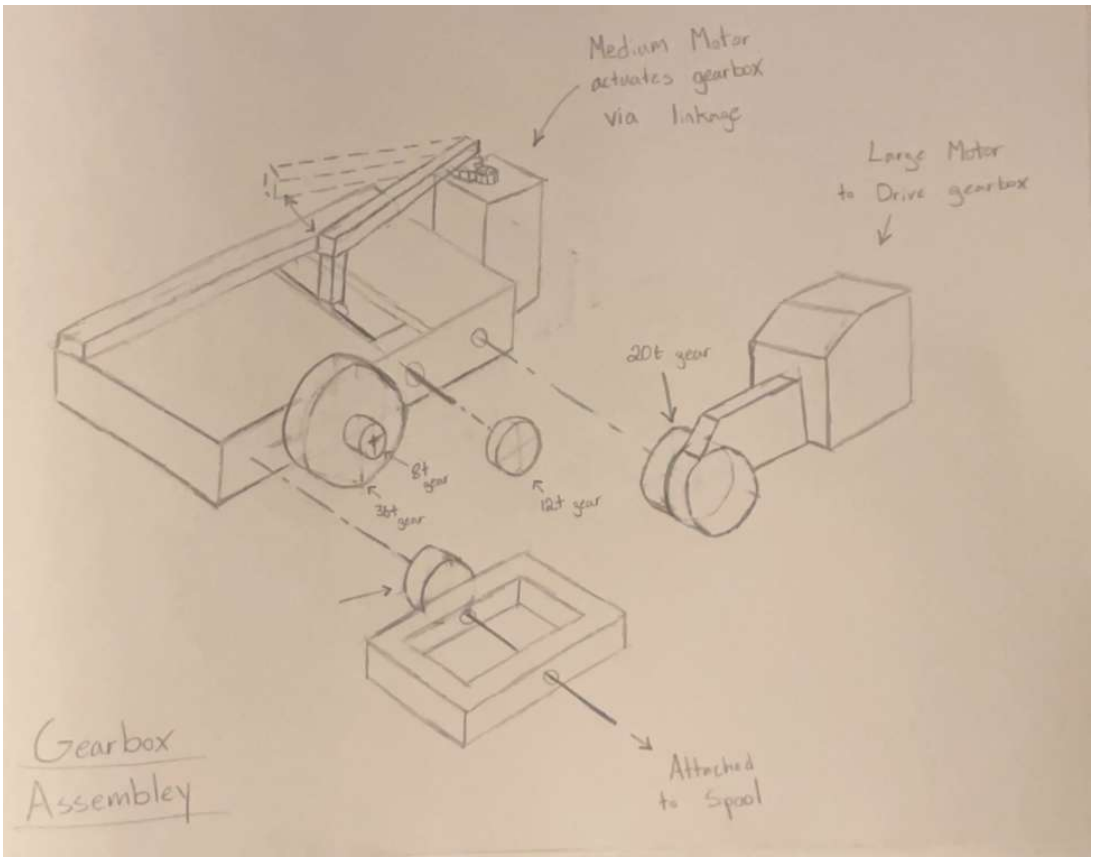

Figure 7: Gearbox First Iteration Freehand

The concept worked, but due to reliability issues with the linkage, the design had to be improved. The second and final iteration replaced the linkage with a rack and pinion

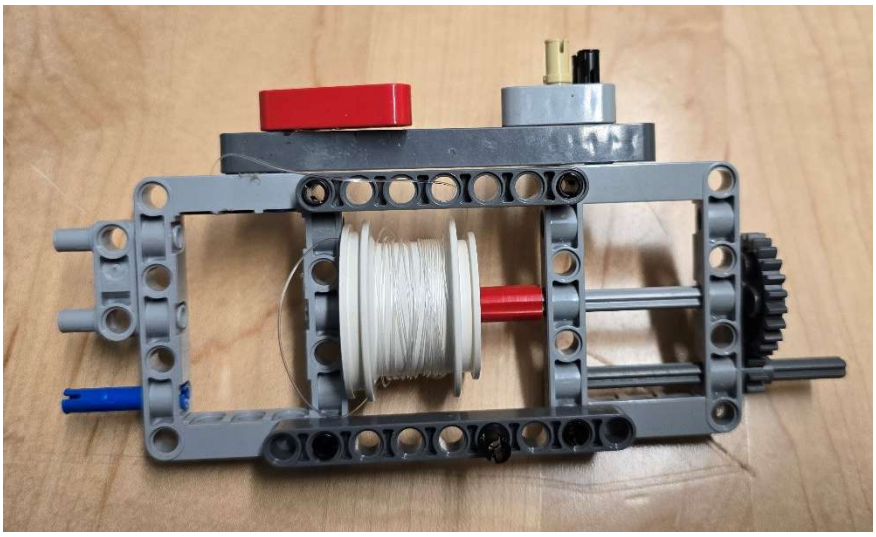

Figure 6: Reel

system. The pinion was driven by the medium motor, which shifted the rack back and forth, engaging and disengaging the idler. The system was successfully and reliably able to shift

between a driven and neutral position, allowing for the casting mechanism to work effectively. The system also included a gear reduction to accommodate for the added torque needed to tension the elastic bands.

4.4 Turret

The first idea for the turret was to allow for lateral movement from the fish and could counterrotate to successfully catch a fish moving in multiple directions. The turret was built using a LEGO rotating platform piece. This piece allowed for a large rotating base but had high friction and poor stability. The small amount of play between the stationary member and the rotating member caused the two pieces to rub together and cause friction. This issue was catalyzed by the large mass imbalance of the arm. The rotating platform was driven by a LEGO large motor with a significant gear reduction. This meant that for the motor to be back driven by a fish, it would have to overcome a large torque reduction and a highly resistive motor. The amount of force from the fishing line needed to

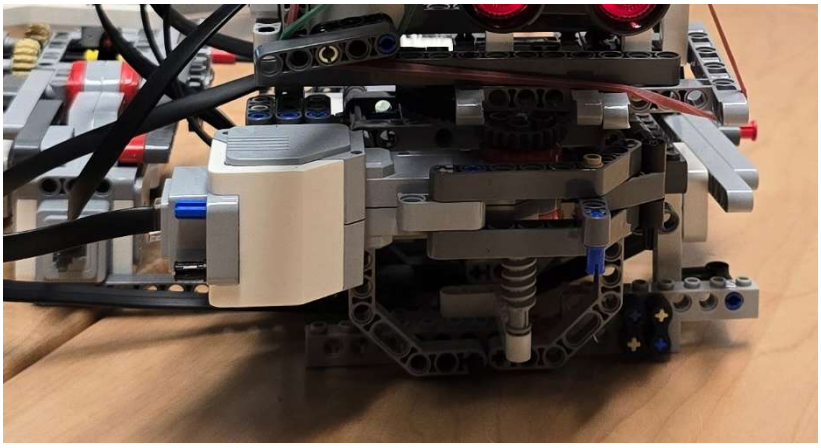

Figure 8: Turret Motor

overcome the resistance and back drive the motor would cause the frame to flex and the geartrain would slip. This issue caused the team to scrap the reactive turret idea and have turret motion purely as a means of automatic-fishing calibration and as a degree of freedom for manual fishing. The first construction of the turret system suffered from slippage and interference issues with the frame below it. The platform was raised by a small amount, and the driving gear and motor was heavily reinforced and constrained to prevent it from moving and slipping. These modifications proved to be very beneficial for the operation of the turret, but there were still instances of slipping due to the inherent flex in the LEGO frame and the high load that the motor was under. The system was successfully able to be moved for the automatic-fishing calibration and manual-fishing control within a range of approximately 90 degrees.

4.5 Arm

The initial idea for the arm was for it to be driven directly by a large motor, however it was determined that due to the weight of the casting mechanism, as well as the force imparted on the arm by the reel, that it would be easy for the motor to be unintentionally back driven, and unable to lift the load of the casting mechanism. Therefore, it was decided to add a worm gear drive to the motor to prevent it from undesirable movement. Due to the amount force, multiplied by the mass of the casting mechanism and leverage radius of the arm, which acted on the worm gear, extra reinforcements had to be constructed around it to prevent it from moving. The strategy for successfully reinforcing the worm gear was to use pieces in such configurations that constrained its vertical movement as best as possible and mechanically locked it in place, rather than purely by friction. The arm elevation system and the casting mechanism were built separately with the intention of later being joined. While a long arm is theoretically helpful in fishing, due to the mechanical limitations of LEGO pieces and powertrains, the team had to limit the length of the arm to ensure that the arm elevation system could effectively elevate the mass of the casting mechanism. Furthermore, the instability of the rotating platform and base meant that there could not be a significant cantilever, thus limiting the length of the arm.

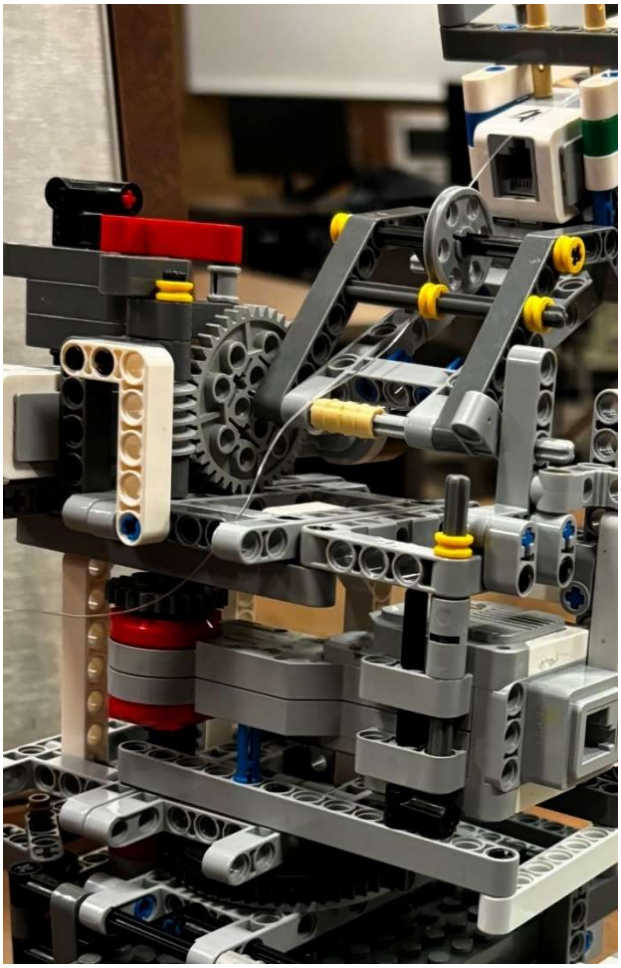

Figure 9: Arm Gearbox

5.0 Software Design and Implementation

5.1 High Level Overview

The codebase of this project is broken down into the startup sequence the three main tasks that the robot should perform, and the shutdown sequence. The main tasks were automatically fishing (autoFish()), manually selecting where to fix (manualFish()), and the user interface (interface()), where the user can choose between the two. Since both autonomous and manual control were set as criterion, the team chose to split the functions as aforementioned.

Since the robot does not require a large amount of data processing, data was stored in the program using local and global bools, which were used to repeat or break out of the wait function, trigger the reel function, or allow the cast system to cast a line. Two timers were

used, one for timing the waitForFish function such that it has a condition that it breaks out of its loop after the period set through its parameter, and a general timer to measure the total runtime of the robot.

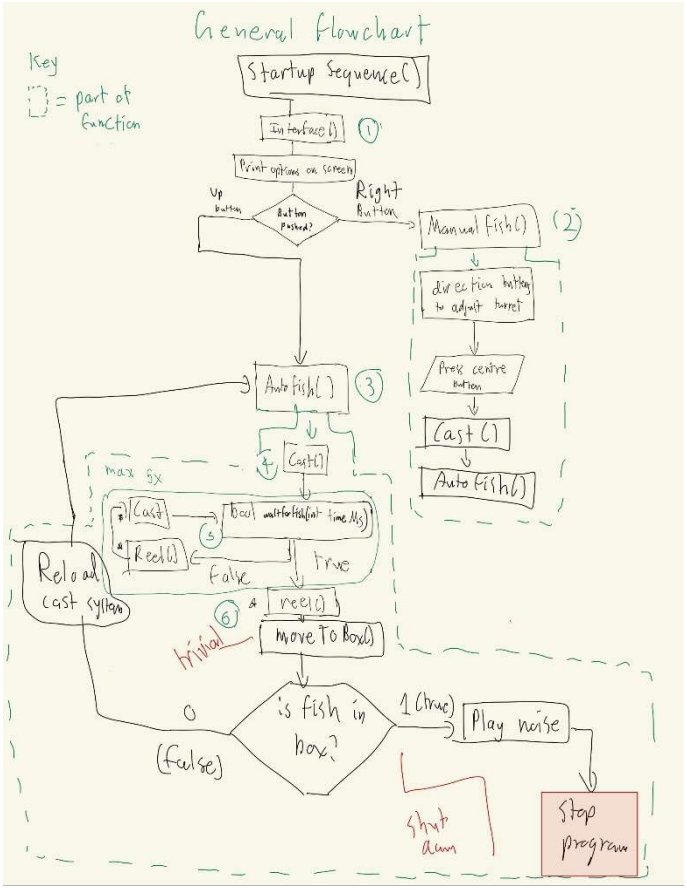

Figure 10: Overall Flowchart

Autofish() contains most of the logic driving the robot, and was further split into cast(), reel(), and waitForFish(), since these were the three actions that the team agreed could summarize the main movements of fishing. A flowchart of the general logic behind the code can be found in appendix B.

5.2 Task List

The task list for the demo was significantly changed due to several issues with the team's design that were discovered well into the robot's development and could not be resolved due to time constraints.

The requirement for the cast to launch at least twenty centimetres was removed because the mechanical design of the casting system required the bait and reel to be reeled tight provide the force to launch the bait. Further detail can be found on section 4.2.

Furthermore, the startup task list was modified from automatically zeroing robot without human intervention. This was because the gyro automatically zeroes every shutdown, so the only way for the robot to be calibrated automatically was to have another touch sensor as the limiter switch. Since EV3s have a maximum of 4 sensor ports and it was initially planned that the robot would use another touch sensor to detect fish landing in a storage bucket, it would require the removal of the ultrasonic sensor to accommodate another touch sensor, removing a key feature of the robot. As a result, this task was altered to just the robot being able to be manually calibrated on startup.

The test of detecting fish on the line was removed since the gear ratios used to connect the reel motor to the reel and line had a large mechanical advantage. This required an extreme pull from the line for the reel motor encoder to detect movement. As a result, this part of the test was modified to just directly turning the reel motor. More detail on this issue is located on section 4.3.

The test of picking up and placing fish in a bucket was removed as the idea of having a fish bucket that the robot dropped fish into was removed due to time constraints.

removed straightening line when fish veers robot. This was again due to the structural problems surrounding the robot arm, explained in further detail on section 4.4.

5.3 Functions

5.3.1 Startup

Written by: Rahul Patel

Parameters: None

Return Type: Void

How it works:

First asks user whether robot is in its startup configuration (perpendicular to platform with robot arm parallel to surface). If it is, user presses the “up” button on the EV3 robot, and the program zeroes the gyros and all the encoders. If the robot is not in position, the user presses the down button, and the program runs the manualFish function, which lets the user adjust the position of the arm with the EV3 direction buttons. Once the desired

starting position is achieved, the user presses the centre button, and the program zeroes the gyro and encoders.

5.3.2 Interface

Written by: Aryaan Ray

Parameters: None

Return Type: bool

How it works:

This function prints out the manual and automatic options for a user to choose with the left and right EV3 buttons. In the main function, the return value of this function is stored as the “IsManual” bool. If the left button is pressed for manual mode, the function returns true, and if the right button is pressed, the function returns false, turning on automatic mode through the main function. The team decided to have this function return a value back to the main function instead of directly running AutoFish() or ManualFish() for easier debugging and code readability.

5.3.3 Cast

Written by: Aryaan Ray

Parameters: None

Return Type: Void

How it works:

The function engages the reel motor with the reel through the gearbox motor, and begins to reel the line, tightening the launching mechanism until the touch sensor is pressed, which stops the motor.

The reel motor is then set to coast mode for its encoder to detect small tugs from the line, and “cast clutch” is then run in reverse, disengaging the reel from the motor and launching the line. Further detail of this mechanical system can be found in section 4.3.

The function uses the ultrasonic sensor to detect whether there is an object closer than fifteen centimetres from the robot. If there is, the robot prints “unsafe” onto the EV3 screen

and unwinds the reel, only allowing the robot to launch once it detects no objects closer than 15 cm to the ultrasonic sensor.

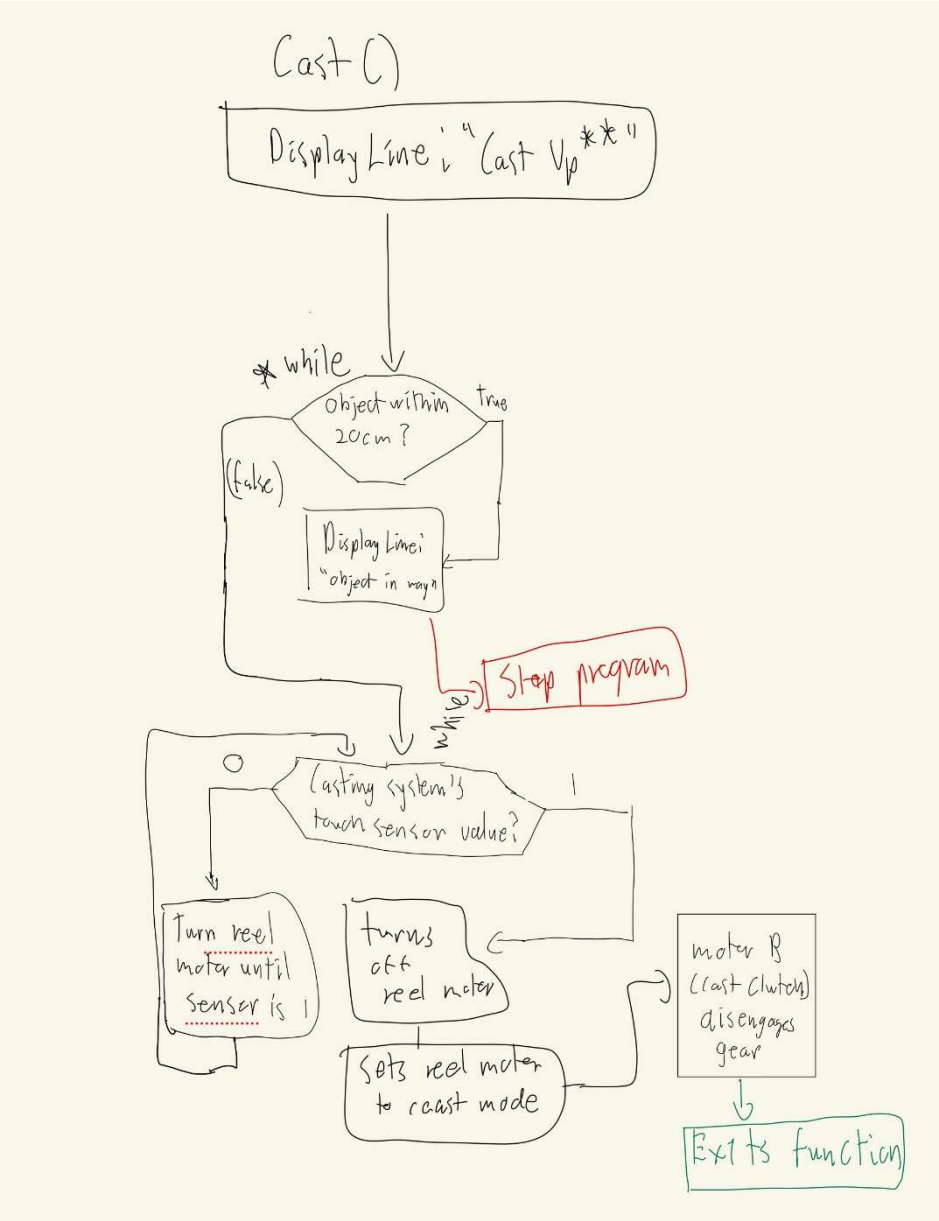

Figure 11: Cast Function Flowchart

5.3.4 waitForFish

Written by: Liam Doyle

Parameters: float

Return Type: bool

How it works:

This function uses the reel motor encoder to detect any tugging on its line, using a timer to wait for a specified duration given by its float parameter. If it detects any change in the encoder's position before the time limit is reached, it returns true. If not, it returns false. The return bool is stored with a local "fishCaught" variable initialized to false, which indicates whether the encoder detected a fish or not. When the encoder detects a fish, this bool is set to true and returned, where it is used by the reelFish and waitForFish functions.

5.3.5 reelFish

Written by: Ian Martin

Parameters: None

Return Type: Void

How it works: This function takes in the fishCaught bool as a parameter returned by the waitForFish function, and if it receives true, it powers the reel motor in reverse until the crossbow touch sensor returns true. It then unwinds the reel to reduce tension on the crossbow system.

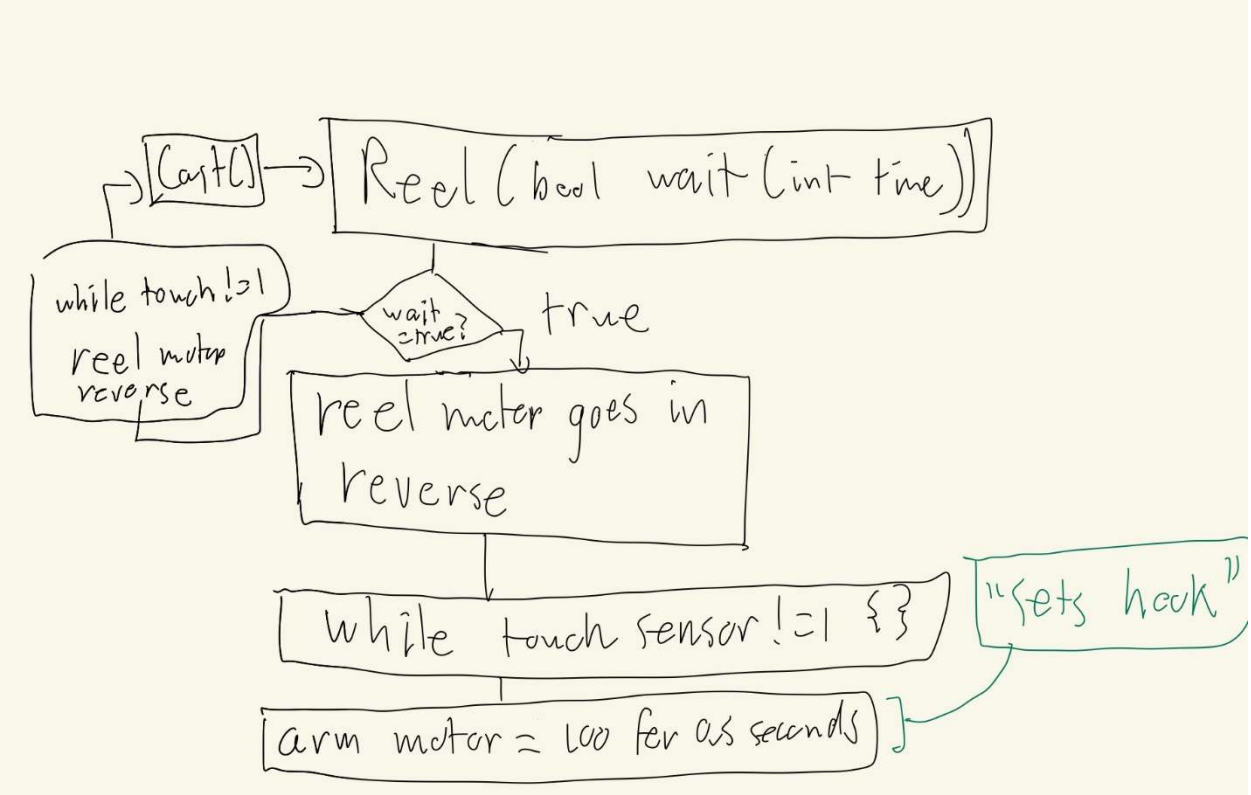

Figure 12: Reel Function Flowchart

5.3.6 AutoFish

Written by: All

Parameters: None

Return Type: Void

How it works:

This function uses a stored bool “test” that stores the return value of the waitForFish function. If the waitForFish function returns a true value, this is inputted into the reelFish() function, where the robot will begin to reel in the fish. If it returns false, the reelFish() function will do nothing due to how it is written, and robot will recast and wait until either the test bool is true (indicating a fish was detected) or until the timer reaches 10 minutes. It will then break out of the loop and exit the function, starting the shutdown sequence.

5.3.7 ManualFish

Written by: Aryaan Ray

Parameters: None

Return Type: Void

How it works:

This function uses direction buttons on EV3 module to move robot arm left and right by powering the swivel motor, and up and down by powering the arm elevation motor. This function is used both for the startup calibration and manual mode selection as it allows the same code to be easily implemented into both scenarios without copying code.

5.3.8 Shutdown

Written by: Rahul Patel

Parameters: None

Return Type: Void

How it works:

This function sets the robot back to its starting position by moving the arm elevation and swivel motors until their encoders read zero, as well as de-tensioning the crossbow on the arm by powering the reel motor until the crossbow touch sensor returns true, and then running the reel motor in reverse for a set period of 2 seconds. It finally prints out the total operating time, as recorded from timer one.

5.4 Testing

The robot was tested in by unit testing their respective modules. The wiring of each component was organized during the concept phase of the robot by drawing out a wiring diagram, which can be viewed in appendix B. After each program was created and tested, the relevant cables for the module wired as per the wiring diagram, and that specific function was tested. For example, for testing the casting module, the cast clutch and reel motor were the only cables connected to the EV3 module when tested. Using this approach, any issues in either the code or the construction of the system would be apparent, since each program was tested in an isolated manner by running each function individually through the main function. As a result, the team was able to quite quickly dub

and mechanical or software issues. Once all the hardware modules and software modules were verified to work, they were all connected into its final assembly and a main function that had all the logic in the robot was created and tested.

5.5 Obstacles to Implementation

Overall, the software implementation for this project went smoothly with minimal issues. The two issues that the team encountered was inputs being used for multiple operations, and a lack of a proper zeroing system. These did impact the final design of the robot, but the team was able to resolve these issues in a way that minimized the number of changes that needed to be made.

The first obstacle that the team encountered was an input being used for multiple functions, leading to issues when operating the robot. For the shutdown process, the enter button was used as an emergency shutoff while the robot is operating, however this button was also used to confirm the user's settings in manual mode. This led to a situation where the user would confirm their settings in manual mode with the enter button and before they could release the button the robot would begin its shutdown procedure. This was resolved by removing the emergency shutoff feature, relying on the other sensors to initiate the shutdown procedure.

The second obstacle that was encountered is the lack of a proper zeroing system. Due to the mechanical design of the robot and the limited number of sensors, there was not an effective way to implement a system that would allow the robot to zero itself. More specifically, the gears driving the turntable for the turret would frequently skip, meaning the encoder could not be reliably used. The team decided to resolve this by implementing a way to calibrate the robot during startup, allowing the user to zero the robot before it begins to operate.

6.0 Verification

Below is the updated list of constraints that was used to evaluate the robot during the demonstration.

1. The turret must have a limited angle of rotation to avoid tangling or putting stress on the cables and fishing line. These limits must also extend to the manual mode to prevent the user from over rotating the robot.
2. The launching mechanism must be light to prevent the center of gravity from shifting too far from the center of rotation for the turret.

3. The robot must be designed to operate without putting stress on the LEGO pieces.
4. The EV3 buttons must be accessible.
5. The reel must be able to spin freely when the robot is casting the lure to allow for slack, but also must be driven by a motor when reeling the lure in.
6. The robot must be able to detect a fish through rotation of the reel motor.

Nearly all these constraints were met during the demonstration as a result of the team's mechanical and software design choices. For the first constraint, the limits were implemented in the software by preventing the motors from running if the encoders or gyro have reached a certain value. This solution eliminated the need for physical hard stops; however, it relied heavily on the robot being properly zeroed before operation. For the second, third, and fourth constraints, all of them were met through the team's approach to the mechanical design of the robot. By designing the robot as subsystems instead of as one singular device, the components were modular, and adjustments could be easily made to the robot without requiring full disassembly. The design was centered around the EV3 brick and expanded outwards, allowing for the buttons to remain accessible, and a number of revisions were made to the arm and launcher designs to reduce their weight. At the demonstration, the robot was able to operate without any support, completing multiple casting cycles without any gear skipping or loose parts. For the fifth constraint, this was created once the team discovered the launcher would not be able to overcome the resistance of the reel motor, even with the motor in coast mode. This constraint was met with the addition of the gearbox mechanism, allowing for the reel to be disconnected from the motor so it could spin freely when necessary. The final constraint is the only one that was not able to be fully met. This is because the gear ratio required for the reel motor to overcome the tension in the launcher's elastic bands resulted in an extreme amount of force being required to rotate the motor by pulling on the fishing line. In testing, the robot would fall over or begin to break apart before the motor could detect rotation. The workaround for the demonstration that the team decided on would be to manually rotate the motor to simulate a fish being detected.

7.0 Project Plan

The project management plan for this project was designed to be as flexible as possible to avoid interfering with other courses. It can be divided into three main parts, leadership, task organization, and time management. In terms of leadership and decision making, each team member had equal say in the project, with disagreements being resolved in a vote. The decision was made to allow the team to put more time into the robot instead of

dealing with a lengthy decision-making process. Another benefit of this leadership style is that each team member was free to work on the components of the project that suited their strengths. As for time management, the initial plan was based around the use of a Gantt chart to schedule task and track progress.

In practice, the project plan allowed the team to devote nearly all the time spent towards the robot instead of administrative tasks. At the beginning of the project each team member chose a subsystem that interested them and became the lead for that aspect. This allowed the team to make progress on multiple subsystems at the same time before coming together to integrate them with each other. However, the success of the plan was reliant on the teams frequent meeting schedule, averaging about three meetings per week. This is because the Gantt chart quickly became outdated, leaving in-person conversations as the primary way the team planned out due dates and progress.

8.0 Conclusions

In conclusion, the final configuration of the robot was able to meet most of its objectives. The robot had both rotational, and elevation control; was able to cast the line a short distance; interrupt the casting process if an object was detected in front of it; and reel the line after a fish was detected. However, the robot struggled with detecting a fish through the motor encoders as there was both too much slip in the geartrain and too much resistance from the motor. This can largely be attributed to the limitations of the Lego EV3 system. It was assumed that the EV3 large motor would have very little resistance when in coast mode, however, this was found to not be the case. Due to this as well as the mechanical losses from the large gear reduction, the reel motor was unable to be rotated by pulling on the line without damaging the robot. Due to this, this had to be demonstrated by manually rotating the reel motor to simulate catching a fish. This same limitation led to the casting mechanism being unable to cast as far as initially planned as the inclusion of the reel gearbox introduced significant mechanical losses that reduced the distance that the line could be cast.

9.0 Recommendations

9.1 Mechanical Design Recommendations

While the team was satisfied overall with the fishing robot, there are several recommendations or changes that the team would have liked to implement. A significant source of issues that the team was constantly addressing was the frame flex and instability

of the build. While LEGO has its structural limitations, the team realized that constructing the base first with appropriate reinforcement to withstand the tension of the fishing line while casting, and the mass of the arm cantilevering off the main structure would have made the whole assembly significantly stronger. The team had to make many structural amendments just to have the robot operational. An alternative solution to constructing the robot in a more structurally-sound way would be to integrate each subsystem in a single, cohesive assembly. This would have meant that each part was closer together, reducing frame flex, and allowing for each member to be structural. The team was not completely satisfied with the performance of the casting mechanism. The cast would often not travel further than a centimetre out of the assembly or get jammed in the carriage rail. This was due to the resistance the lure would have to overcome to pull the fishing line out of the reel, and the obstruction the design of the carriage rail provided. An alternate design the team considered that would not need a long arm was a flywheel mechanism. The team's success with the clutch mechanism, combined with the energy storage of a flywheel, provided a hypothetical solution that could theoretically launch the lure with a higher degree of success. The clutch mechanism could have been used to quickly transfer energy from the flywheel to a launching mechanism. While an obvious method to track rotation of the turret would be to use the turret motor encoder, the team opted for a gyro sensor instead. While the encoder was considered, it was deemed too inconsistent due to the potential of the turret gears skipping, causing the turret to lose its center position. Since a gyro sensor detects rotation relative to its ground position, geartrain skipping is not a problem, and thus provides an ideal solution.

9.2 Software Design Recommendations

While the team was largely satisfied with the software design and implementation for the robot, there were some items that the team felt could have been improved. One of these was the lack of software limits on the vertical motion of the arm. This made it possible for the arm to keep running even after reaching its hard stops and creating the potential for the arm to damage them with its high torque.

Appendix A - Code

```
void cast() //Aryaan Ray
{

      SensorType[S4] = sensorEV3_Touch;
      SensorType[S2] = sensorEV3_Ultrasonic

      motor[motorB] = -20;
      wait1Msec(1000);
      motor[motorB] = 0;
      bool cast = 0;
      while(!cast)
      {
      while(SensorValue[S4] != 1)
      {
            motor[motorD] = 50;
      }
      motor[motorD] = 0;


            if(SensorValue[s2] > 15)
            {
            setMotorBrakeMode(motorD, motorCoast);
            motor[motorB] = 20;
            wait1Msec(400);
            motor[motorB] = 0;
            wait1Msec(1000)
            motor[motorB] = -20;
            wait1Msec(500);
            motor[motorB] = 0;
            wait1Msec(1000);
            displaybigTextLine(3, "")
            cast = 1;
            }
            else
            {
                  displayBigTextLine(3, "Unsafe")
                  motor[motorD] = 100;
       while(SensorValue[S4] == 0)
       {}
       motor[motorD] = 0;
       motor[motorD] = -100;
       wait1Msec(2000);
       motor[motorD] = 0;
```

```
            }
      }
}

bool waitForFish(float timeLimit) // Liam Doyle
{
      // Reset Variables
      bool fishCaught = false;
      setMotorBrakeMode(motorD, motorCoast);
      clearTimer(T2);
      nMotorEncoder[motorD] = 0;

      while((time1(T2)/1000) != 5){}
  // Buffer period to allow lure to settle on table

      while(!fishCaught && (time1(T2)/1000) <= timeLimit){
  // Poll for encoder movement or time limit
            if(nMotorEncoder[motorD] != 0){
                  fishCaught = true;
            }
      }
      return fishCaught;
}


void reelFish(bool fishCaught) //Ian Martin
{
  if(fishCaught)
  {
        motor[motorC] = 100;
        wait1Msec(1000);
        motor[motorC] = 0;
  }
  motor[motorD] = 100;
  while(SensorValue[S4] == 0) {}
  motor[motorD] = 0;
  motor[motorD] = -50;
  wait1Msec(1000);
  motor[motorD] = 0;
}

void manualFish() //Aryaan Ray
{
      eraseDisplay();
      displayBigTextLine(3,"Manual Mode");
      displayTextLine(5,"Press Enter when done");
```

```
      wait1Msec(100);
      while(!getButtonPress(buttonEnter))
      {
            motor[motorC] = 0;
            motor[motorA] = 0;
        while (getButtonPress(buttonUp))
            {
                  motor[motorC] = 100;
            }

            while (getButtonPress(buttonDown))
            {
                  motor[motorC] = -100;
            }

            while (getButtonPress(buttonLeft)&& getGyroDegrees(S3) > -
40)
            {
                  motor[motorA] = -20;
            }

            while (getButtonPress(buttonRight)&& getGyroDegrees(S3) <
40)
            {
                  motor[motorA] = 20;
            }
            eraseDisplay();
      }

}

void shutDown() //Rahul Patel
{
       while(nMotorEncoder[motorC]!=0)
       {
            if (nMotorEncoder[motorC]>0)
            {
            motor[motorC] = -100;
            }
            else
            {
            motor[motorC] = 100;
            }
       }

       motor[motorC] = 0;
```

```
	while((getGyroDegrees(S3)) != 0)
	{
		if(getGyroDegrees(S3) < 0)
		{
			motor[motorA] = 10;
		}
		else
		{
			motor[motorA] = -10;
		}
	}
	motor[motorA] = 0;
	motor[motorD] = 100;
	while(SensorValue[S4] == 0)
	{}
	motor[motorD] = 0;
	motor[motorD] = -100;
	wait1Msec(2000);
	motor[motorD] = 0;

	displayTextLine(4, "Operating Time:  %f", time1(T1));
}

void AutoFish() //ALL
{
	eraseDisplay();
	displayBigTextLine(1,"Robot running...");
	bool test = false;
	while (!test && (time1(T1) <= 100000)
	{
		cast();
		test = waitForFish(10);
		reelFish(test);
	}
	eraseDisplay();
	if (test == true)
	{
		displayBigTextLine(1,"Fish caught :DDD");
	}
	else
	{
		displayBigTextLine(1,"Fish not caught :(");
	}
}

bool interface() // Aryaan Ray
```

```
{
	//displays all available options
	displayBigTextLine(1,"FishyWishy :D");
	displayTextLine(4,"Please choose an option");
	displayTextLine(5,"with the direction buttons");
	displayBigTextLine(7,"AutoFish: Right");
	displayBigTextLine(9,"Manual: Left");
while(!getButtonPress(buttonLeft) && !getButtonPress(buttonRight)) {}
	if(getButtonPress(buttonLeft))
	{
		while(getButtonPress(buttonLeft)) {}
		//clear screen and says reeling mode
		return true;

	}

	else if(getButtonPress(buttonRight))
	{
		while(getButtonPress(buttonRight)) {}
		//clear screen and says reeling mode
		return false;

	}
	else
	{
		displayTextLine(1,"wrong input, defaulted to autofish");
		wait1Msec(1000);
		return false;
	}

}

void startUp() //Rahul Patel
{
	eraseDisplay();
  displayTextLine(4,"Is the robot in its startup configuration?");
  displayTextLine(5,"Up for Yes, Down for No");
  while(!getButtonPress(buttonAny))
  {}
  if(getButtonPress(buttonUp))
  {
	nMotorEncoder[motorA] = 0;
	nMotorEncoder[motorC] = 0;
	resetGyro(S3);
  }
  else if(getButtonPress(buttonDown))
```

```
  {
     eraseDisplay();
    displayTextLine(4,"Please adjust robot");
    displayTextLine(5,"to startup config");
    displayTextLine(6,"Press Enter when complete");
    while(!getButtonPress(buttonEnter))
    {
     displayTextLine(2,"manualFish");
          manualFish();
    }
    nMotorEncoder[motorA] = 0;
    nMotorEncoder[motorC] = 0;
    resetGyro(S3);
  }

  clearTimer(T1);
}

task main()
{
     //base of menu system
     SensorType[S3] = sensorEV3_Gyro;
     wait1Msec(50);

     SensorMode[S3] = modeEV3Gyro_Calibration;
     wait1Msec(50);

          resetGyro(S3);
          wait1Msec(50);

          startUp();
          bool isManual = interface();

          if(isManual)
          {
               manualFish();
          }
          AutoFish();

     eraseDisplay();
     displayBigTextLine(1, "Shutting down...");
     shutDown();
}
```