Alex M., Aras V., Tobi O., Tim M.,
Michael W., Aiden B.

ENGR 7B: Introduction to Engineering

University of California, Irvine

Winter 2024

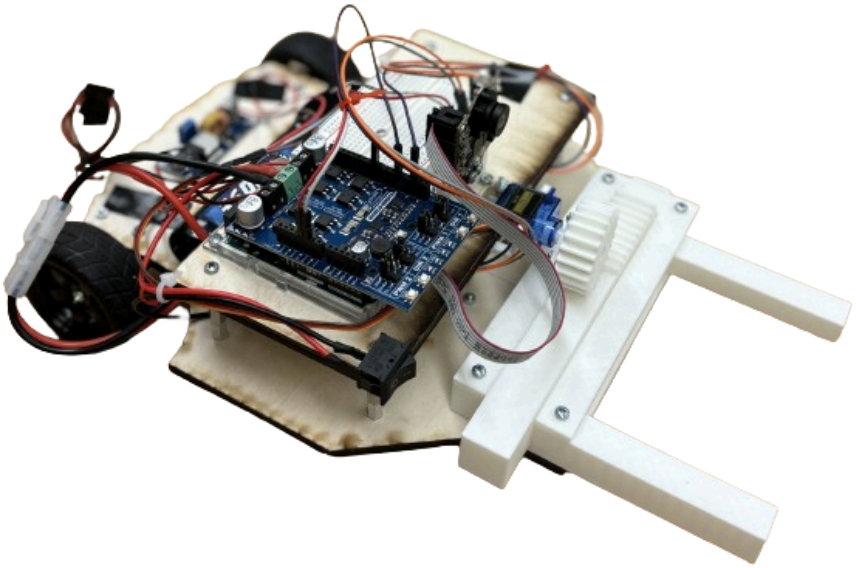


Table of Contents

Executive Summary....3

Problem Definition....4

- Introduction....4
- Technical Review / Background....4
- Design Requirements....6

Design Description....7

- Summary of Design....7
- Design Details....8
- Wiring Diagram....9
- Algorithm Design....10

Action Item Report....16

- Task Assignment....16
- Gantt Chart....18

Evaluation....20

- Calculations....20
- Test Plan....21
- Results & Discussion....22

Appendix A: SOLIDWORKS Drawings....25

Appendix B: Bill of Materials....36

Appendix C: Arduino Autonomous Mapping Code....37

Appendix D: References....44

Executive Summary

ENGR 7B constitutes the latter portion of the two-quarter Introduction to Engineering class catered to UCI engineering Freshman. Its primary aim is to instruct students in the diverse processes and disciplines associated with developing an autonomous rover capable of line following and, ultimately, tracking and retrieving a can at its designated endpoint. At the onset of the quarter, group formation was facilitated, leading to the establishment of our 6-member team comprising students from various engineering disciplines, collectively known as “Team RhynO”. Leveraging our shared experiences from Engr. 7A in the previous quarter, each member brought unique skills to the table.

The project kicked off with the practical application of lecture and lab insights into our weekly action plans. Beginning with the 3D CAD design modeling of the rover and its components using SolidWorks, we proceeded to estimate costs and compile information for the purchase order form. Subsequently, attention turned to the implementation of safety protocols and procedures essential for the physical fabrication of the chassis and other wooden components. This was followed by the meticulous soldering and positioning of electronics. Finally, the testing and evaluation phase rounded off the project. Regular group meetings convened every Tuesday from 5 pm to 6 pm, during which assigned roles guided each member's contributions both during sessions and beyond.

Naturally, the iterative nature of the project necessitated numerous revisions and adjustments to tackle encountered challenges. Issues such as inadequate light reception by IR sensors, deviations from the desired line trajectory, and missed can retrievals were among those addressed. Collaborative efforts during labs and weekly meetings facilitated the resolution of these issues. Adjusting the placement of IR sensors proved effective in rectifying light reception problems, while updates to the Arduino code enabled successful can retrieve and precise line following.

Drawing inspiration from the mechanisms employed by previous successful rovers in the course, our design featured a straightforward claw mechanism executing a horizontal closing motion for can retrieval. Each team member's strengths contributed significantly to crafting a rover that not only adhered to prescribed guidelines but also showcased the collective creativity of the team. Ultimately, the end product exemplified this fusion of ingenuity and technical proficiency, fulfilling its intended objectives.

Problem Definition

Introduction

The problem we were presented with and tasked to solve was designing and fabricating an automated rover that could follow along a path on the floor, properly detect a specific colored object, and move toward then grasp the detected object. Within the task at hand also came further restrictions, mainly being that of limited budget and time availability, as we could only work on our rover during specific lab hours. With the three main goals of the project considered, our objective was to be able to quickly traverse the specified path, efficiently detect the correct object to grasp, and reliably grip the determined object. Furthermore, this was the most simplified version of our objective, because as time passed, our objective adapted to include the issue at hand, such as smoothing out the turns of the line-tracking, as well as detecting the object at a variety of distances and angles. Overall, the efficiency and reliability of our rover was our priority, as we would not be able to implement the trial and error process if our rover was unable to consistently succeed in the specific tasks given to us.

Technical Review / Background

The most notable part of the rover that we created was undoubtedly the concept of mimicking full autonomy through the use of line following sensors. To gain an understanding of how line following technology works, and the cause of their conception, we conducted research on the matter. The idea of line-following robots dates as far back as the 1960s. The second World War gave birth to a period of innovation that was fueled by the necessity to win the war for both sides. This period happened to spill over to following decades, which saw the birth of computer science as an academic discipline. Processing power was not yet at the extent to allow full autonomous movement as so line-following algorithms were conceived to test guided autonomous movement. Today this technology is mainly used in educational settings and occasionally more seriously in manufacturing and warehousing. While not as stimulating as understanding guided autonomy, the use of the Pixy2 for object detection is important to highlight. Pixy Cams got their start in 2013 and are fast vision sensors for DIY robotics and similar applications; we used it for object detection.

In relation to our rover design, the key takeaway from our research was that the concept behind a

line-following robot involves employing sensors to identify the line and subsequently utilizing a controller to regulate both the speed and direction of the robot, ensuring it remains aligned with the line. In our case, the "controller" was the written program that the Arduino, the brain of our rover, would relay to the motors. Given that our goal was to produce a competitive rover, in terms of track time, this meant optimizing our code to be easily changed and purely operational focused. As such, we realized that the most simple yet effective way to turn our rover was to leverage our control of motor speeds. With our naviance programming skill set, the most realistic route was to program the robot to check for the black line using the IR sensors and make point turns and swing turns, accordingly. In a point turn both wheels turn, but in opposite directions which causes the rover to turn. In a swing turn, only one wheel turns and the other acts as an anchor taking advantage of friction, almost like using a drawing compass. Our programming algorithm used both of these turn concepts to impersonate gradual turns, as we simply did not want to hassle ourselves with more trial and error to get the perfect speeds for gradual turns. In terms of the Pixy2, we used logic and reason to realize that it should be placed on the vertical centroid axis of our claw mechanism to have the center of the displayed image be aligned with it. After identifying the location, the use of the Pixy2 came down to calculating the range of x-coordinates that the can should cover in order to engage the closing of the claw. In simpler words, we combine math with trial and error to see how close we should be to the can for the rover to stop and grab it.

References:

Wikipedia contributors. "Mobile Robot." Wikipedia, 15 Mar. 2024, en.wikipedia.org/wiki/Mobile_robot

FutureLearn. "Updates, Insights, and News From FutureLearn | Online Learning for You." FutureLearn, 25 Oct. 2022, www.futurelearn.com/info/courses/robotics-with-raspberry-pi/0/steps/75898

Wikipedia contributors. "Computer Science." Wikipedia, 23 Jan. 2024, en.wikipedia.org/wiki/Computer_science

Turning. robocatz.com/turning.htm

Design Requirements

Size requirements:

(a) Rover must be smaller than 12x16 inches

Structure requirements

(a) Easily accessible battery switch

(b) Battery must be easily removable

(c) No use of internal combustion engines

Safety requirements

(a) No sharp objects that protrude out

(b) All wires and connectors are insulated and covered

Cost requirements

(a) Total cost of the rover must be less than $300

Product Deliverables

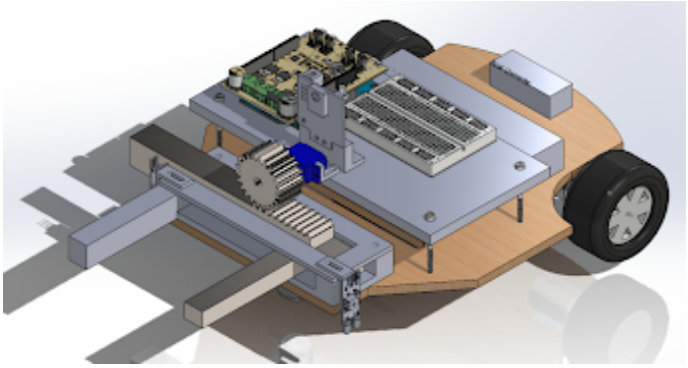
(a) The produced rover must be able to follow a black line and grab a detected object.

(b) The final design report must include the finalized bill of materials, parts list, Gantt chart, and drawings of the rover.

(c) A final oral presentation that demonstrates a business plan for our rover in under 5 minutes.

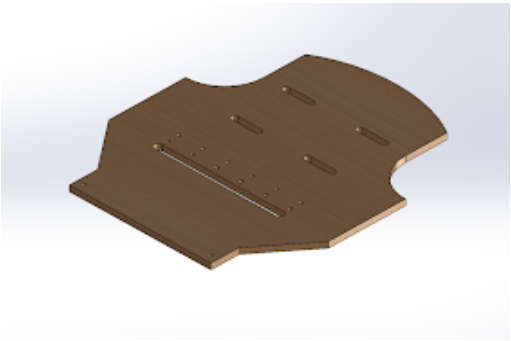
Design Description

Summary of Design



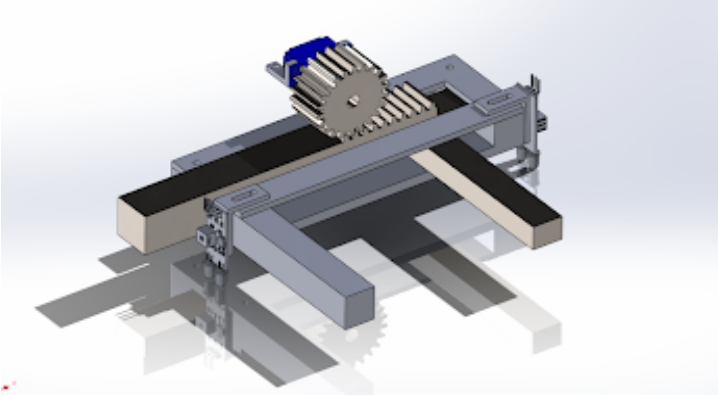
The final design of our rover is pictured to the right. It includes the gripper, the drive train, the sensors, and the electronics all connected by a wooden chassis. The gripper is at the front of the rover, and it is designed to grab objects detected by the PixyCam. The drivetrain is in the back and controls the direction and the speed of the back wheels, which also control the direction and speed of the entire rover using a 2 wheel directional steering mechanism.

The Arduino, breadboard, and motor shield are on the lofted piece of wood to allow for ease of adjustment. The battery is placed in the back because it is the heaviest, and it evens the weight from the gripper and electronics to keep the rover stable. The battery is attached to the chassis via velcro straps that go through holes cutout in the chassis. The buck converter (also in the back) is attached to the chassis with 2 M3 screws. The ESC, On/Off switch, and receiver are attached to the chassis with double-sided tape. The Arduino is close to the breadboard so it is able to be easily wired, and the same is true for the buck converter. This allowed wire organization to be simple. The IR sensors are in the front to allow for early detection of the line and a well lit detection placement. The PixyCam is centered on the robot and at the top of the loft, to allow us to see past the gripper.

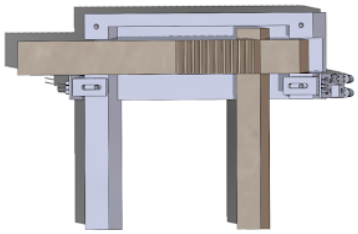


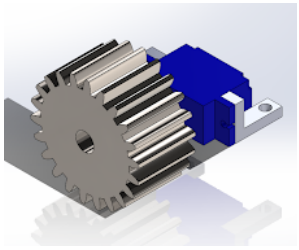
We chose to use 65 mm wheels, as they were the smallest available wheels for this project and allowed for easy drivetrain adjustment. The small wheels made changing motor speed less dramatic and allowed for finer overall movement. The overall dimensions for the entire rover came out to be 12.54 inches by 10 inches, which fits the requirements that the size needs to be within 16 inches by 10 inches.

Design Details

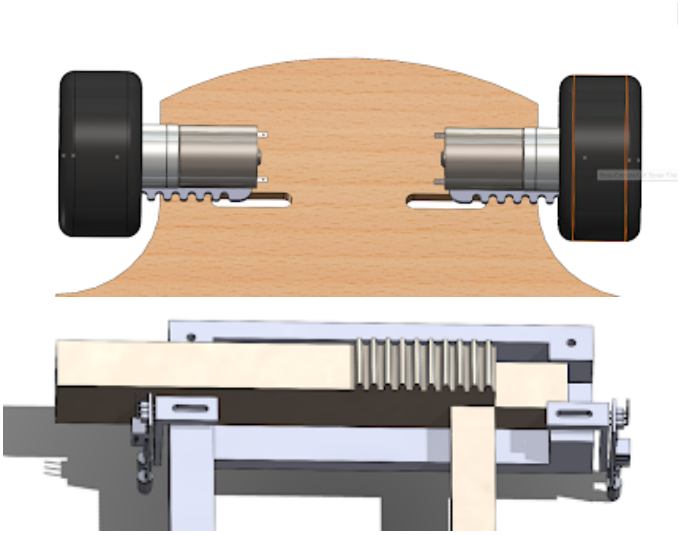


For the gripper, we decided to go with a 3d-printed design to allow for fine adjustment of the parts. We used a rack and pinion system to translate the circular motion of a servo into horizontal movement of the gripper claws. The servo was mounted on the front of the loft using 3d-printed mounts and M3 screws. Attached to the servo is the Pinion Gear (pictured below). Below the pinion gear is the rack, which has the moving gripper finger attached to it. This rack is placed into the Gripper Base (pictured right), which provides the second gripper finger and controls the rack's movement. The rack also has pieces extending outward from it to ensure that it stays sturdy in the Gipper Base. The Gripper Top keeps the Base and rack together. These are all mounted at the front of the Chassis using M3 screws. The width of the gripper opening is 4 inches, and it closes 1.6 inches to graph the 2.5 inch can. It extends 3 inches from the front of the Chassis.



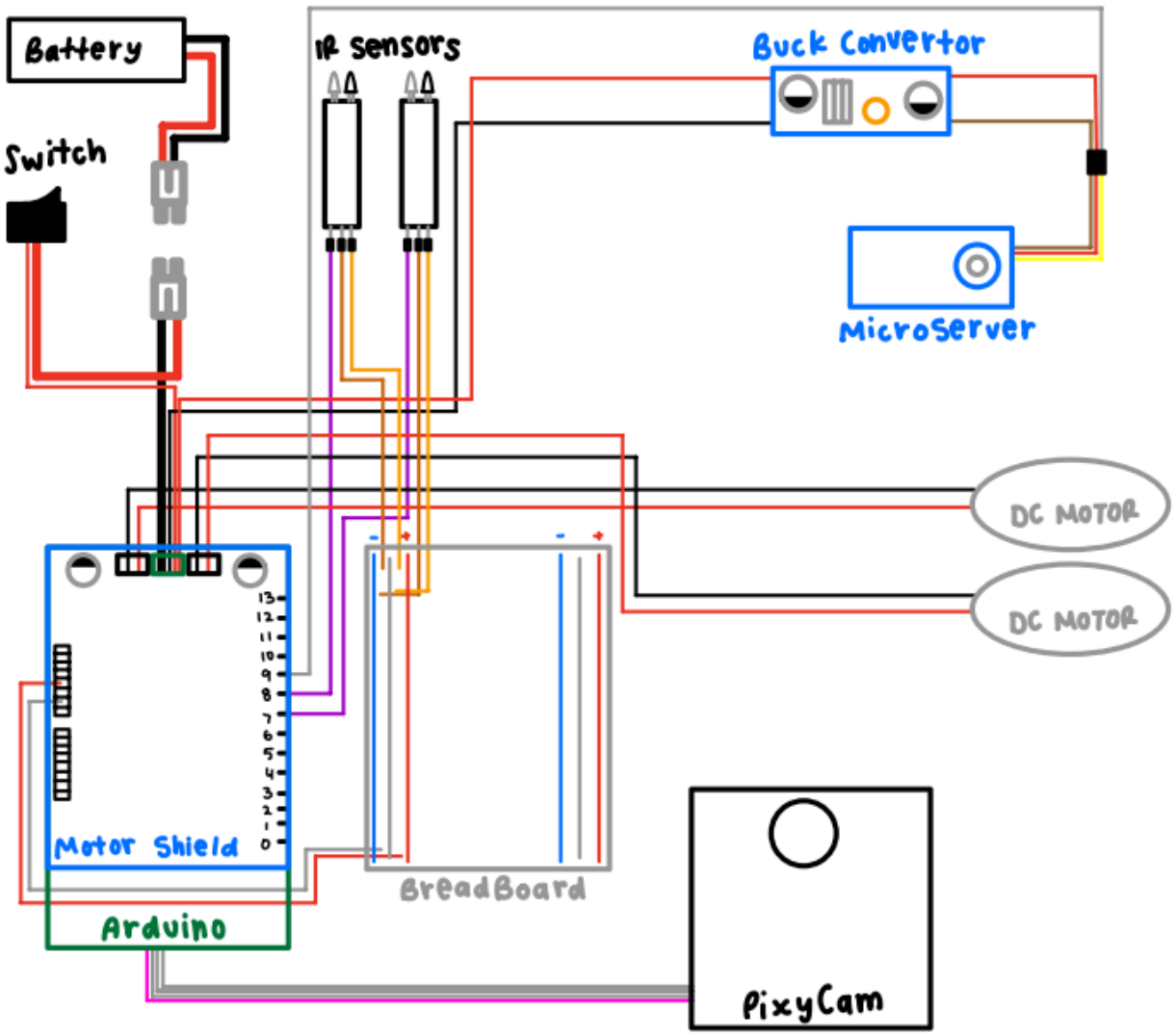


The drivetrain and steering is powered from the back (pictured below). The drive train is made up of two motors that both spin forwards and backwards. There is a caster wheel in the front to allow the robot to move in any direction. The rover steers by adjusting the motor speed. When the motor wants to turn in a direction, the motor on that side slows down and the motor on the opposite side speeds up. This allows the rover to turn whatever side it needs to. The steering is guided by the IR sensors and Pixy Cam in the front. When the IR sensors detect the line, they tell the motors to steer the rover in the



opposite direction. We mounted the IR sensors to the edge of Gripper (pictured above) to allow for good lighting and early detection. Then, when the PixyCam detects the can, the Rover drives towards it.

Wiring Diagram



The diagram provided is hand drawn and illustrates the comprehensive wiring network of our project. It encompasses various electronic components including a **micro servomotor, battery, buck converter, motors, PixyCam, Arduino, motor shield, breadboard, and IR sensors.**

Powering the motor shield and consequently the entire rover is a 3600 mAh, 7.2 V battery. **Serving**

as the central processing unit, the Arduino hosts the code necessary for the rover to execute line tracking functionalities. Positioned atop the Arduino, the motor shield facilitates differential motor control.

The motors are directly connected with the motor shield. A buck converter is employed to regulate the battery's voltage from 7.2 V to a consistent 5 V, supplying power to the micro servo motor responsible for the rover's grabbing mechanism.

The PixyCam is directly connected to the Arduino to identify and track the red can, enabling the rover to approach and grasp it at the designated destination. IR sensors, integrated with both the breadboard and motor shield, aid in line tracking and course correction.

Lastly, a switch, connected to the motor shield, initiates the rover's journey upon closure.

Algorithm Design

For the coding portion of this project, the group was given the task of utilizing an Arduino board to automate a rover such that it would be able to: traverse a line-drawn path, utilize PixyCam detection to locate a can, and collect the can between a claw system. The logic for this process was split into 3 separate sections. First, the rover needed to follow a line-drawn path. To complete this task, the rover utilized readings it gained from two IR sensors to make adjustments which allowed it to maintain a trajectory that followed the lines. Once the rover reached the end of the course, it would switch to the second section of its logic. For this section, the rover utilized readings it collected from the Pixy camera to center the can, relative to the rover's claws. When the can was fully centered, the rover would shift to the third stage of its programming. In this stage, the rover was told to move forward until the can was within reach, in which case the rover would close its claws on the can and cease movement.

The code was written in an order which followed this three step process. Starting with the line following and, using the conditions mentioned prior, continuing to the next section of the logic. The rover was also coded to only center the can once, since this would optimize claw retrieval speed.

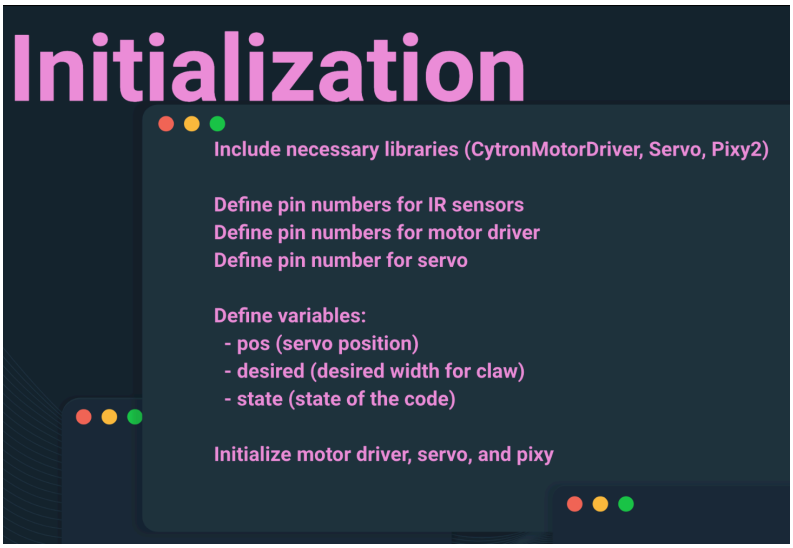
Below is the Arduino and pseudocode for which was written in the process of programming the rover. The code is segmented and displayed in the order it was written, from top to bottom:

Initialization:

Arduino Code

```
// Calling Claw, Servo and Pixy Libraries
#include <CytronMotorDriver.h>
#include <Servo.h>
#include <Pixy2.h>


// Inititializing IR Pins
#define R_S 2   // IR sensor Right
#define L_S 5   // IR sensor Left



// Configure the motor driver.
CytronMD motor1(PWM_DIR, 3, 4);  // PWM 1 = Pin 3, DIR 1 = Pin 4. Left
CytronMD motor2(PWM_DIR, 6, 7); // PWM 2 = Pin 6, DIR 2 = Pin 7. Right



Servo Claw;  // Creates servo object to control claw



int pos = 180;    // Variable to store the servo position
int desired = 130; // variable to find desired width
int state = 1; // variable to define the state of the code




Pixy2 pixy; // Creates pixy objext to control pixy2 camera
```

Pseudocode

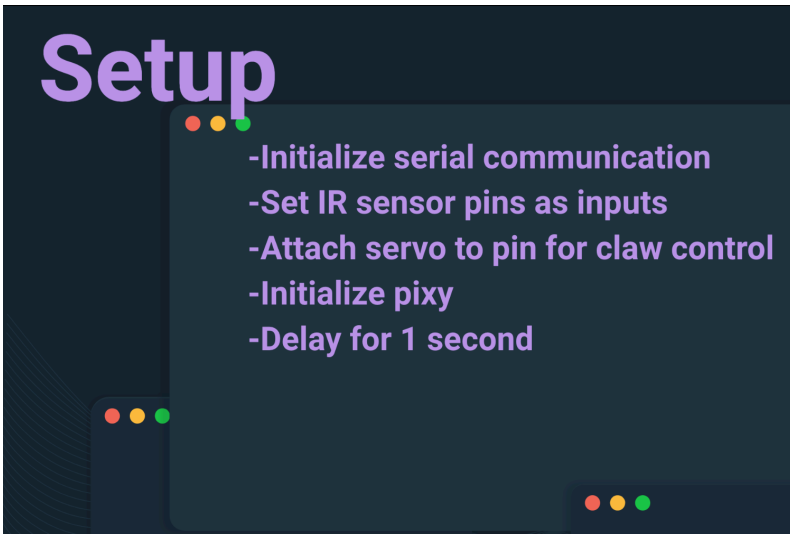


Setup:

Arduino Code

```
// Sets up a couple things for rover

void setup()
{
  Serial.begin(9600);

  //Set IR Sensors as pins
  pinMode(R_S, INPUT);
  pinMode(L_S, INPUT);

  // Attach servo to pin 9 and set claw to open
  Claw.attach(9);
  Claw.write(pos);

  //Initialize pixy
  pixy.init();

  delay(1000);
}
```

Pseudocode

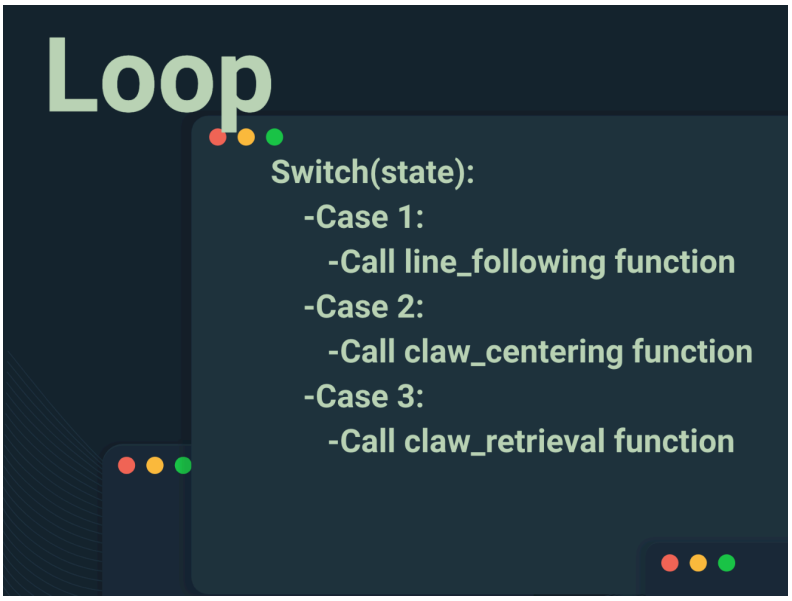


Loop(Switch Cases):

Arduino Code

```
81  void loop()
82  {
83    switch(state)
84    {
85      case 1:
86      line_following();
87      break;
88
89      case 2:
90      claw_centering();
91      break;
92
93      case 3:
94      claw_retrieval();
95      break;
96    }
```

Pseudocode

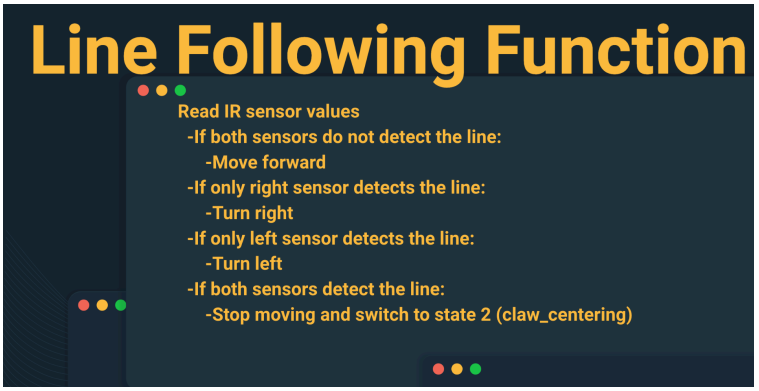


Loop(Switch Cases):

Arduino Code

```
113 void line_following()
114 {
115
116   if (digitalRead(R_S) == 0 && digitalRead(L_S) == 0) // If neither sensor detects anything
117   {
118     // Rover moves forward
119     forward();
120     delay(3);
121   }
122   else if (digitalRead(R_S) == 1 && digitalRead(L_S) == 0) // If the right sensor detects the line
123   {
124     // Rover turns right
125     turnRight();
126     delay(15);
127   }
128   else if (digitalRead(R_S) == 0 && digitalRead(L_S) == 1) // If the left sensor detects the line
129   {
130     // Rover turns left
131     turnLeft();
132     delay(15);
133   }
134   else if (digitalRead(R_S) == 1 && digitalRead(L_S) == 1) // If both sensors detect the line
135   {
136     // Rover stops moving
137     stop();
138
139     // Rover switches states and begins centering the can
140     state=2;
141   }
142 }
```

Pseudocode

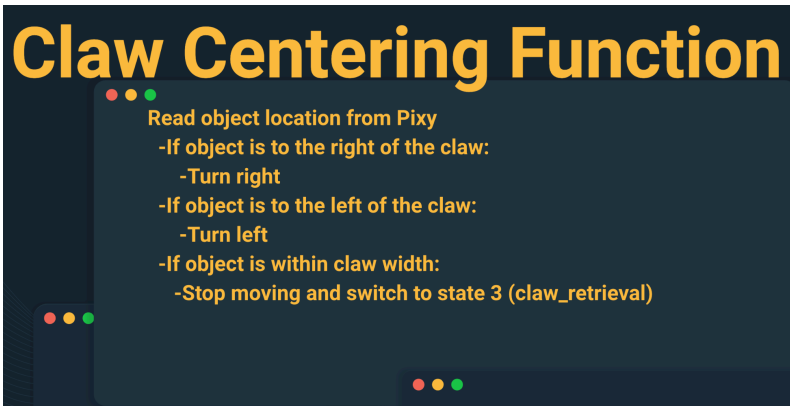


Claw Centering Function:

Arduino Code

```
void claw_centering()
{
  // Print information about can location
  pixy.ccc.getBlocks();
  int object_x = pixy.ccc.blocks[0].m_x;
  int object_width = pixy.ccc.blocks[0].m_width;
  Serial.println(object_x);


  if (object_x < 185) // If can is to the right of the claw
  {
   // Turn Right
    motor1.setSpeed(64);
    motor2.setSpeed(-128);
  }
  else if (object_x > 195) // If can is to the left of the claw
  {
   // Turn Left
    motor1.setSpeed(-128);
    motor2.setSpeed(64);
  }
  else // If can is within claw width
  {
   // Rover stops
   motor1.setSpeed(0);
   motor2.setSpeed(0);

   // Switch case to claw retrieval function
   state=3;
  }
  delay(100);
}
```

Pseudocode

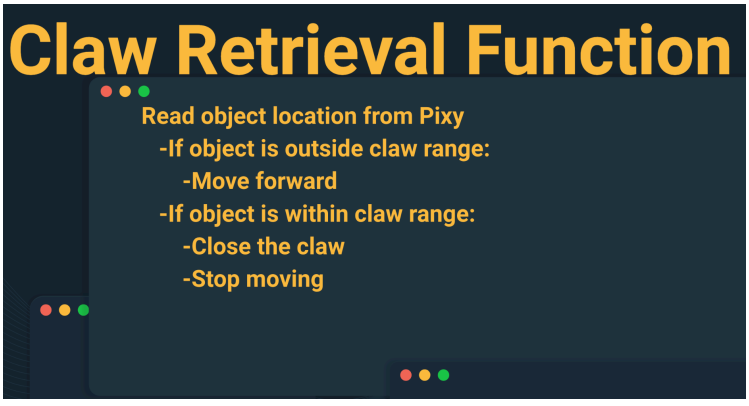


Claw Retrieval Function:

Arduino Code

```
void claw_retrieval()
{
  // Print information about the can location
  pixy.ccc.getBlocks();
  int object_x = pixy.ccc.blocks[0].m_x;
  int object_width = pixy.ccc.blocks[0].m_width;
  Serial.println(object_x);

    if (object_width < desired) // If can is outside claw range
    {
      // Rover goes forward
      motor1.setSpeed(-250);
      motor2.setSpeed(-250);
    }

    else if (object_width >= desired) // If can is in range of claw
    {
      // Claw closes
      Claw.write(80);

      // Rover stops
      motor1.setSpeed(0);
      motor2.setSpeed(0);
    }
  delay(10);
}
```

Pseudocode

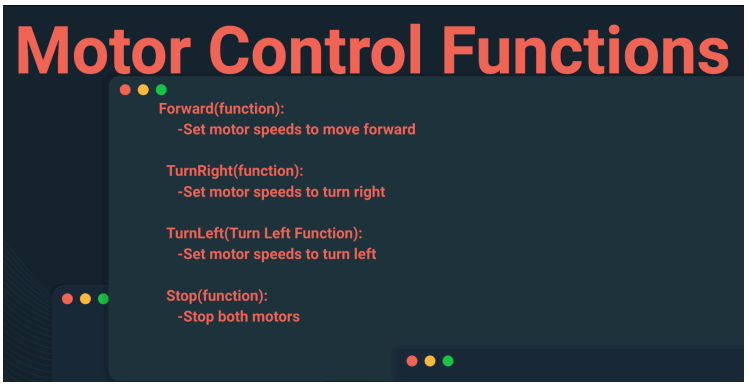


Motor Control Functions:

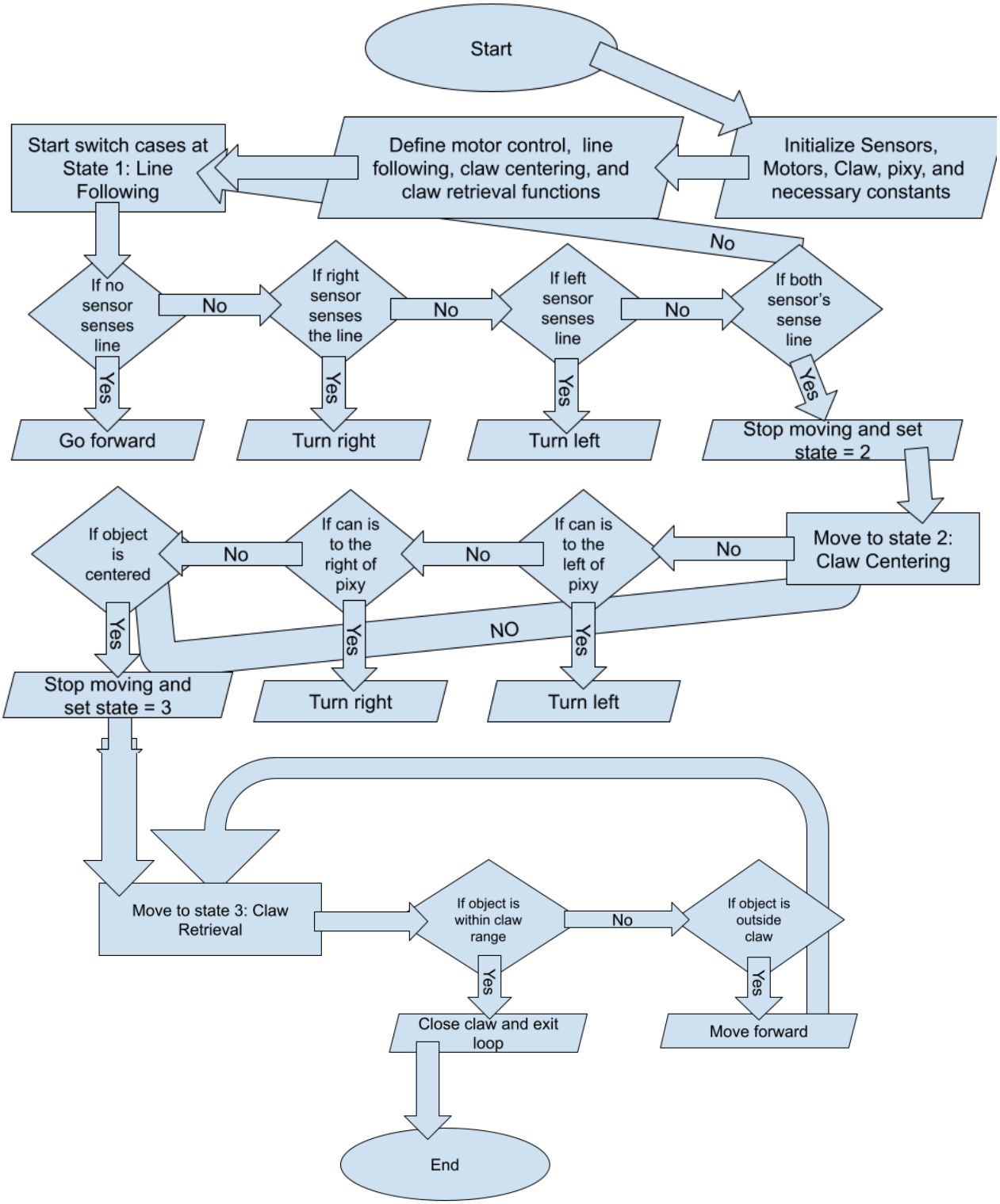
Arduino Code

```
//Motor Control Functions – (For the line following code)

void forward() // Makes Rover go forward
{
  motor1.setSpeed(-150);   // Motor 1 continues forward at 50% speed.
  motor2.setSpeed(-150);   // Motor 2 continues forward at 50% speed.
}


void turnRight() // Makes Rover turn Right
{
  motor1.setSpeed(256);   // Motor 1 (Left) runs forward at 50% speed.
  motor2.setSpeed(-180);   // Motor 2 (Right) set at -25% speed.
}


void turnLeft() // Makes Rover turn left
{
  motor1.setSpeed(-180);   // Motor 1 (Left) set at -25% speed.
  motor2.setSpeed(256);   // Motor 2 (Right) runs forward at 50% speed.
}


void stop()  // Stops Rover
{
  motor1.setSpeed(0);   // Motor 1 (Left) stops.
  motor2.setSpeed(0);   // Motor 2 (Right) stops.
}
```

Pseudocode



Flowchart:


Start
Initialize Sensors, Motors, Claw, pixy, and necessary constants
Define motor control, line following, claw centering, and claw retrieval functions
Start switch cases at State 1: Line Following
No
If no sensor senses line
No
If right sensor senses the line
No
If left sensor senses line
No
If both sensor's sense line
Yes
Yes
Yes
Yes
Go forward
Turn right
Turn left
Stop moving and set state = 2
Move to state 2: Claw Centering
If object is centered
No
If can is to the right of pixy
No
If can is to the left of pixy
No
Yes
Yes
Yes
NO
Stop moving and set state = 3
Turn right
Turn left
Move to state 3: Claw Retrieval
If object is within claw range
No
If object is outside claw
Yes
Yes
Close claw and exit loop
Move forward
End

Action Item Report

Task Assignment

Each week during the quarter (besides Week 1), we met on Tuesday from 5-6pm at the most convenient and appropriate location. In Week 2, each member of the team was assigned a role as listed below.

- Team Captain: Alex
- CAD Design: Aras, Aiden
- Arduino: Tobi, Tim
- Fabrication: Alex, Michael

We treated the team member who was in charge of each respective job as the leader in getting their job done. For example, if Tobi needed help completing the Arduino, it was his job to delegate the tasks to other team members. This way, although everyone was assigned one job, everyone had an opportunity to partake in every task.

The following is a written version of the Gantt Chart for each week:

- **Week 2**:
    - Assigned Roles (Team Captain, CAD Design, Arduino, Fabrication)
    - Created Team Name: Team RhynO
    - Begin designing rover on SolidWorks

- **Week 3**:
    - Created Gantt Chart
    - Designed Chassis in SDW
    - Continued designing claw mechanism

- **Week 4:**
    - Decided on Gear Ratio
    - Created PO Form
    - **PO Form Due**
    - **Final SDW Design Due**

- **Week 5:**
    - Designed wiring diagram
    - Began soldering wiring
    - Began 3-d Printing & Laser Cutting for chassis
- **Week 6:**
    - Continued 3-d printing & laser cutting
    - Finished chassis fabrication
    - All electronics put on chassis

- **Week 7**:
    - Finished rover fabrication
    - Worked on Arduino code
    - **Preliminary Presentation Slides Due**

- **Week 8:**
    - Tested and adjusted Arduino code
    - **Rover Structure Deadline**

- **Week 9:**
    - Continued testing
        - Got a time of 16.61 seconds

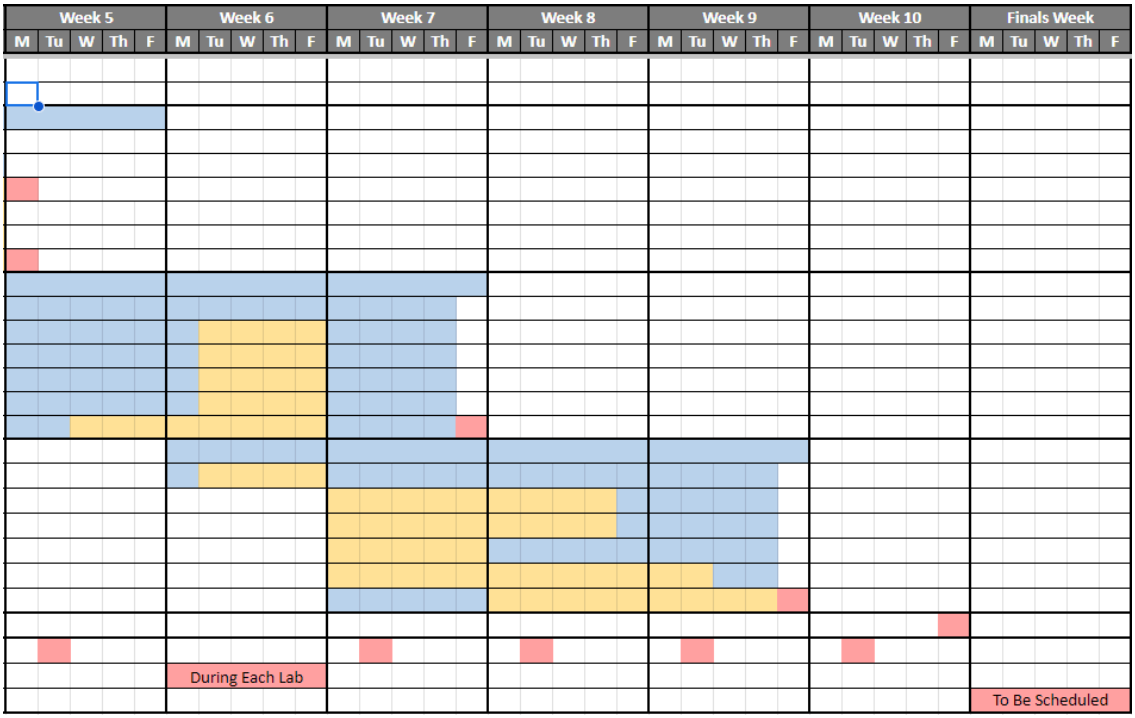
- **Week 10:**

- Continued testing code and attempted to make top ten

- **Finals Week**
  - **Final presentation, video, and design report due on day of presentation (3/19 @ 4pm)**

Gantt Chart

Team RhynO					Planned					Actual					Due Date																
Thursday, 3:00-4:50pm	Planned		Actual		Week 1					Week 2					Week 3					Week 4										
Activity	Start	End	Start	End	M	Tu	W	Th	F	M	Tu	W	Th	F	M	Tu	W	Th	F	M	Tu	W	Th	F						
Team Formation																														
Team Name & Captain Chosen																														
Rover Design																														
Determine Rover Layout																														
Select Gear Ratio																														
Purchase Order Form																														
SolidWorks Part Designs																														
SolidWorks Assembly																														
SolidWorks Detailed Drawings																														
Structure Fabrication																														
Wheels System																														
Motor Mount																														
Gear Guard																														
Battery holder																														
Steering System																														
Structure Assembly																														
Electrical System																														
Mount & Connect Electronics																														
Arduino Tuning for Motor																														
Arduino Tuning for Servo																														
Steering System Testing																														
Claw Mechanism Testing																														
Final Test and Evaluation																														
FINAL COMPETITION																														
Action Item Reports																														
Preliminary Presentation																														
Final Presentation																														


Week 5
M Tu W Th F
Week 6
M Tu W Th F
Week 7
M Tu W Th F
Week 8
M Tu W Th F
Week 9
M Tu W Th F
Week 10
M Tu W Th F
Finals Week
M Tu W Th F
During Each Lab
To Be Scheduled

Evaluation

Calculations

Estimated Weight of Vehicle:

$$Weight \approx \Sigma(W_{Each\ component}) = W_{Battery} + W_{Chassis} + W_{Arduino} + 2W_{Motor} + W_{Servo} + W_{Buck\ Converter} + \ldots$$
$$\ldots + W_{Wheels} + W_{Motor\ Shield} + W_{3D\ Printing} + W_{PixyCam} + W_{Breadboard}$$

$$Weight \approx 0.367\,Kg + 0.2\,Kg + 0.025\,Kg + 2(0.082)\,Kg + 0.009\,Kg + 0.016\,Kg + 0.017\,Kg + \ldots$$
$$\ldots + 0.036\,Kg + 0.07\,Kg + 0.01Kg + 0.079\,Kg = \textbf{\textit{0.993 Kg}}$$

The actual weight of the Rover is 1.12 Kg.

Predicted drive time:

$$DC\ Motor\ Stall\ Current = \frac{(Battery\ Voltage)}{(12V/5A)} = \frac{7.2V}{(12V/5A)} = 3A$$

$$Micro\ Servo\ Stall\ Current = 1A$$

$$Drive\ Time = \frac{(Battery\ Capacity)}{(DC\ Motor\ Stall\ Current)+(Micro\ Servo\ Stall\ Current)} = \frac{3.6Ah}{3A+1A} = 0.9\ Hours = \textbf{\textit{54 Minutes}}$$

Stall Torque:

$$Stall\ Torque = Max\ Stall\ Torque * Voltage\ Conversion = 11\ kg.cm * (7.2V/12V) = \ldots$$
$$\ldots = 6.6\ kg * cm\ for\ each\ motor$$

Max Stall Weight:

$$Max\ Stall\ Weight = Torque / Wheel\ Radius = (0.066\ kg * m) / (0.065\ m) = 1.015\ kg$$

Discussion of Mechanical Advantage and Design Choices:

Our group's choice of the 34:1 gear motor over the 47:1 gear motor relies solely on the concept of mechanical advantage. Mechanical advantage is traditionally defined as the ratio of output force to input force in a system. In the discussion of gears, mechanical advantage is entirely dependent on the gear ratio. A gear ratio can increase the output torque or output speed of a mechanism, but not both. A larger gear ratio will have more torque and less speed, while a smaller gear ratio will have more speed and less torque. Given that the object of this course was to traverse a flat track in the smallest amount of time possible, our decision for the gear ratio was simple: more speed. As a result, we chose the 34:1 gear motor, as the internal gear drive train would provide our rover with greater linear speed than that of the 47:1 gear motor. If the track included obstacles that required the rover to maneuver in an irregular track, then we would have to put more consideration in the torque that our motors would provide, but lucky for us this was not the case. In fact, for this exact reason, we were not too concerned with the stall torque of the 34:1 gear motor. We knew that if we were under the maximum stall weight of the motor, then on flat ground the motor would provide ample torque. The maximum stall weight is the maximum weight the motor can lift. This value was calculated to be: 1.015 kg. We anticipated that the weight of our rover would be 0.993 kg based on calculations, but neglect for miscellaneous items put us at a measured weight of 1.12 kg. Despite this, the motor worked just fine in testing and the competition.

Test Plan

From the beginning of the quarter, our team established a goal, the goal being to complete the physical design and construction of our rover by Week 7 to leave more time for testing. Subsequently, due to the team's diligence and consistency with regard to staying on task and following the Gantt chart, we were able to finish our rover on time, leaving three weeks for logistical testing and optimization.

Of our beginning testing stages was the testing of the **infrared sensors**. This was arguably the most important test prior to the full run through the course, given that the rover could not operate and drive autonomously if it could not follow the lines on the ground to guide it. Unfortunately, whilst navigating the initial testing of the IR Sensors, we encountered an issue with the shadow that our rover's chassis was emitting. The shadow was blocking the visibility of the IR sensors, ultimately restricting our rover from following the line underneath it. This resulted in a few test runs on practice tape where the rover simply

wouldn't run or would quickly go off the track. With purchase orders already submitted and no LED's ordered on our end, we were left with a difficult dilemma. However, with the help of the TA's input and some critical thinking, we decided that the best way to approach this problem was to further space the IR sensors out underneath the chassis and curve the sensors inward in order to increase light intake. This approach served to be successful, and the rover was able to follow the line with ample accuracy.

Following the testing of the sensors came our analysis of the **Arduino code**. The first question was whether we should use **swing** or **point** turns to maximize the efficiency of our rover. Ultimately, we decided that the best option would be to utilize swing AND point turns in different parts of the code. In doing so, the rover could address smoother turns with accuracy and speed, while also having the ability to successfully navigate sharp turns. Next came the testing of the **speed and delay time**. Despite our previous efforts IR sensor testing to allow the rover to follow the line sufficiently, this action could not be completed unless the speed and delay time of the rover were calibrated to the path of the track. Due to the sharp, pointed turns on the track, the rover had to go a certain speed and have a certain delay time in order for the rover to correctly move around the track. Thus, we were able to correlate each aspect accordingly and allow our rover to move.

Another aspect of our rover that we had to test was the efficiency of the gripper. The gripper was ultimately operated by a micro servo connected to the Arduino, activating the gripper once the can was within a certain distance from the PixyCam. Our gripper was one of the best parts of our rover and operated with efficiency, despite previous struggles with the size of the gripper itself.

Lastly, we were tasked with testing the most stubborn part of the code, which was the connection to the **PixyCam**. One of the problems we struggled with was the ability of the PixyCam to identify the can and align the rover with the center of the can in the coordinate system. To satisfy the issue and make our rover more efficient in centering itself after the completion of the lined track, we updated the coordinates for which the can should be positioned to activate a lined drive to the can. Additionally, we changed the sensitivity of the PixyCam to more accurately intake the color red of the can.

The process of testing the rover was a long and tedious process. However, the poise of the group to stay focused and address issues with unity allowed our rover to pull through and ultimately place high on the list of ENGR 7B teams.

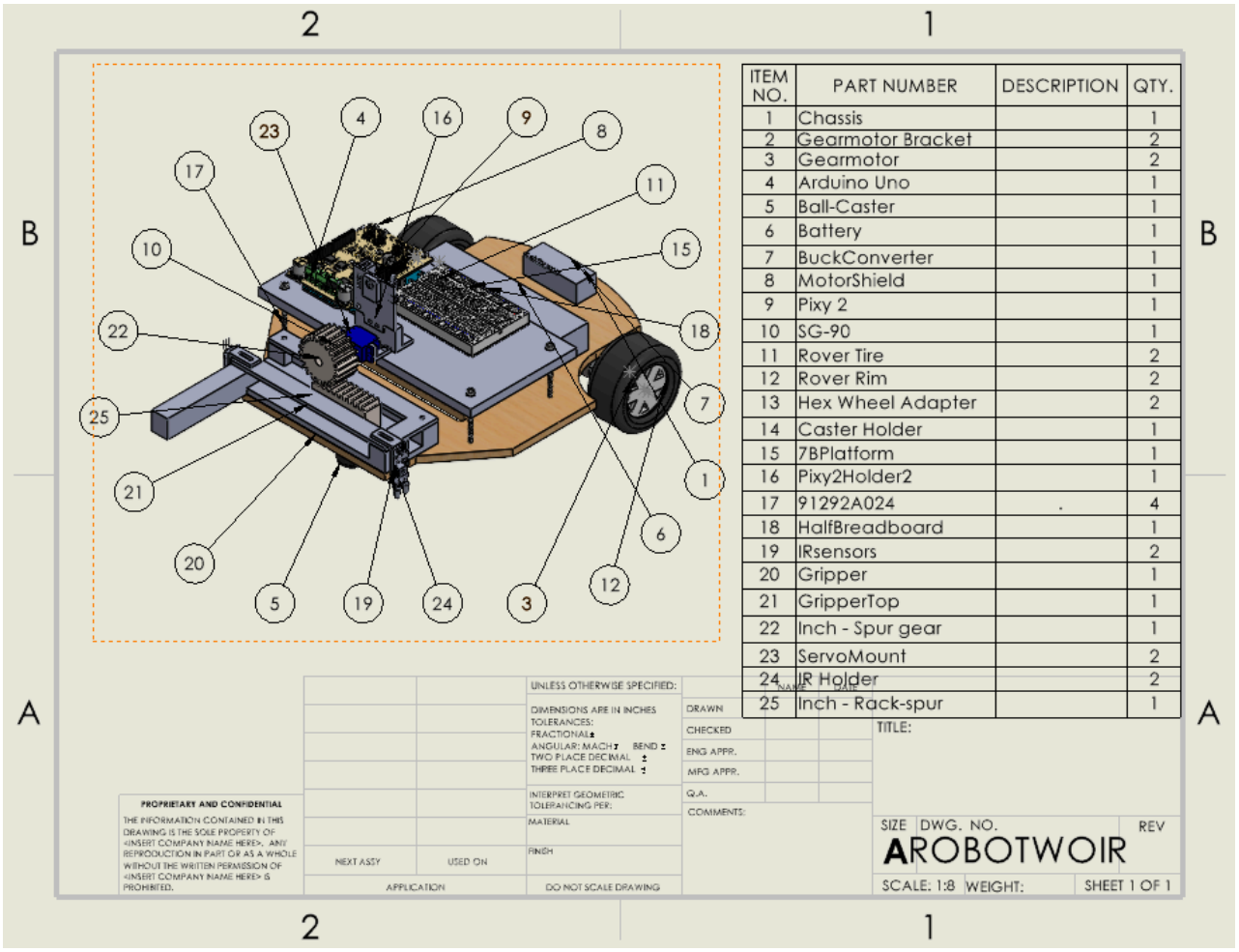
Results & Discussion

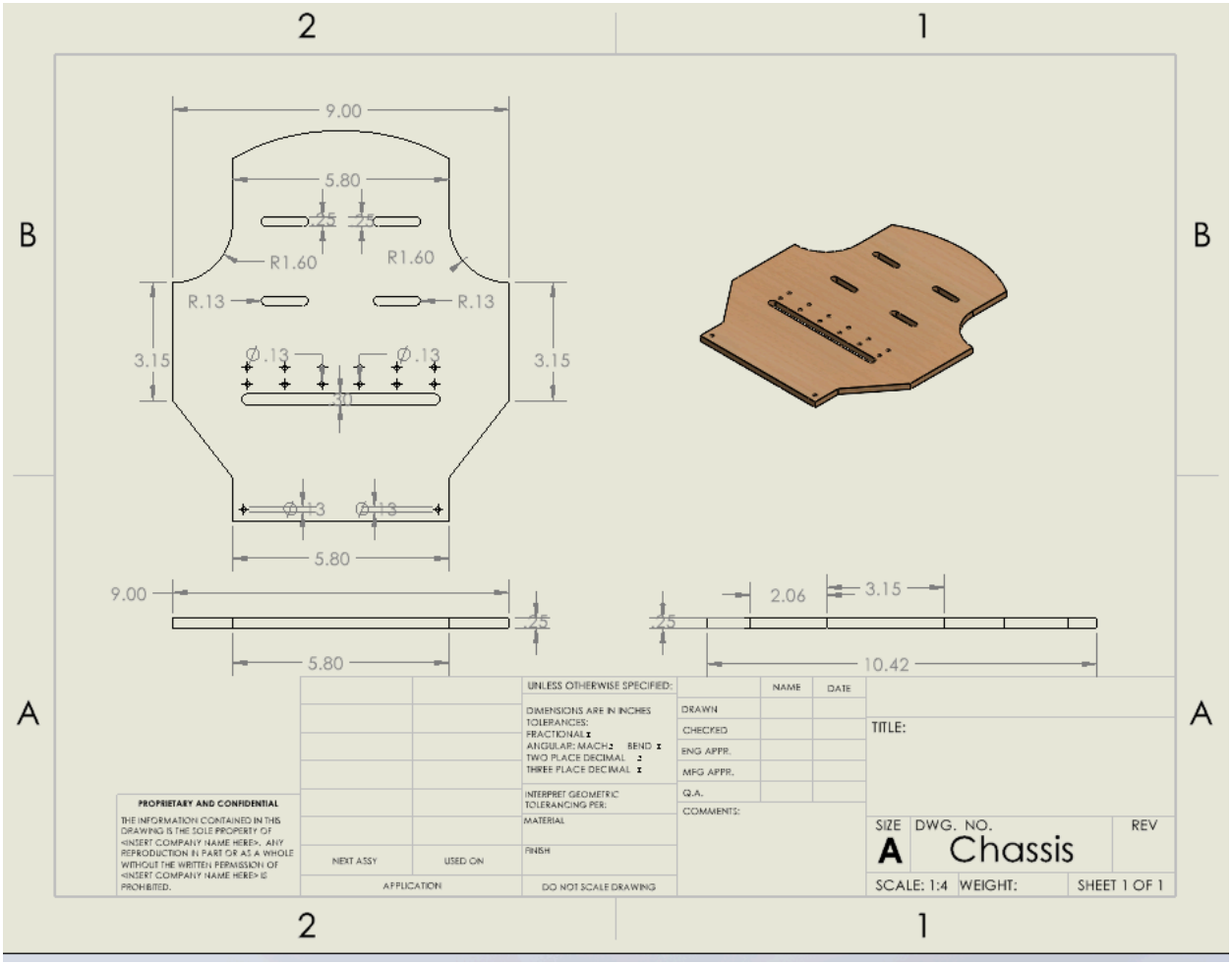
On the basis of our results, we analyzed the evaluation criteria of our rover. First was the stability of our rover. The rover was ultimately very stable as a result of the two wheels in the back of the rover and the caster wheel that held the front of the rover up. Our rover's weight was unbalanced in relation to the wheels being in the back of the rover, and as a result of the caster wheel connected to the caster wheel holder, the rover was able to stay stable despite quick turns that might have caused the rover to fall. As for the weight aspect, our rover stood the test of high weight capacity due to the decision to construct the chassis out of wood. With the abundance of electrical parts to allow the rover to operate, the choice to use a wooden chassis served to be a good one as it allowed the rover to manage the weight distribution successfully. Next came the agility of the rover. Due to the difference between this quarter's competition guideline and last quarter's agility was not the main focus of the rover. However, we did our best to make our rover agile for the means of the track this quarter. This meant correcting the speed based on the sharpness of the turns and maximizing the speed on straightaways. Our control over the rover's agility was on par and led us to the fourteenth spot on the leaderboard. Lastly, our rover stayed durable due to the density of our parts. Each and every part that we either 3D printed or crafted using laser cutting was made to withstand the long process of testing. This allowed us to get our best recorded time on the track near the end of our testing period. Even though much time had passed since the rover's structure had been built, the rover still operated with efficiency and durability.

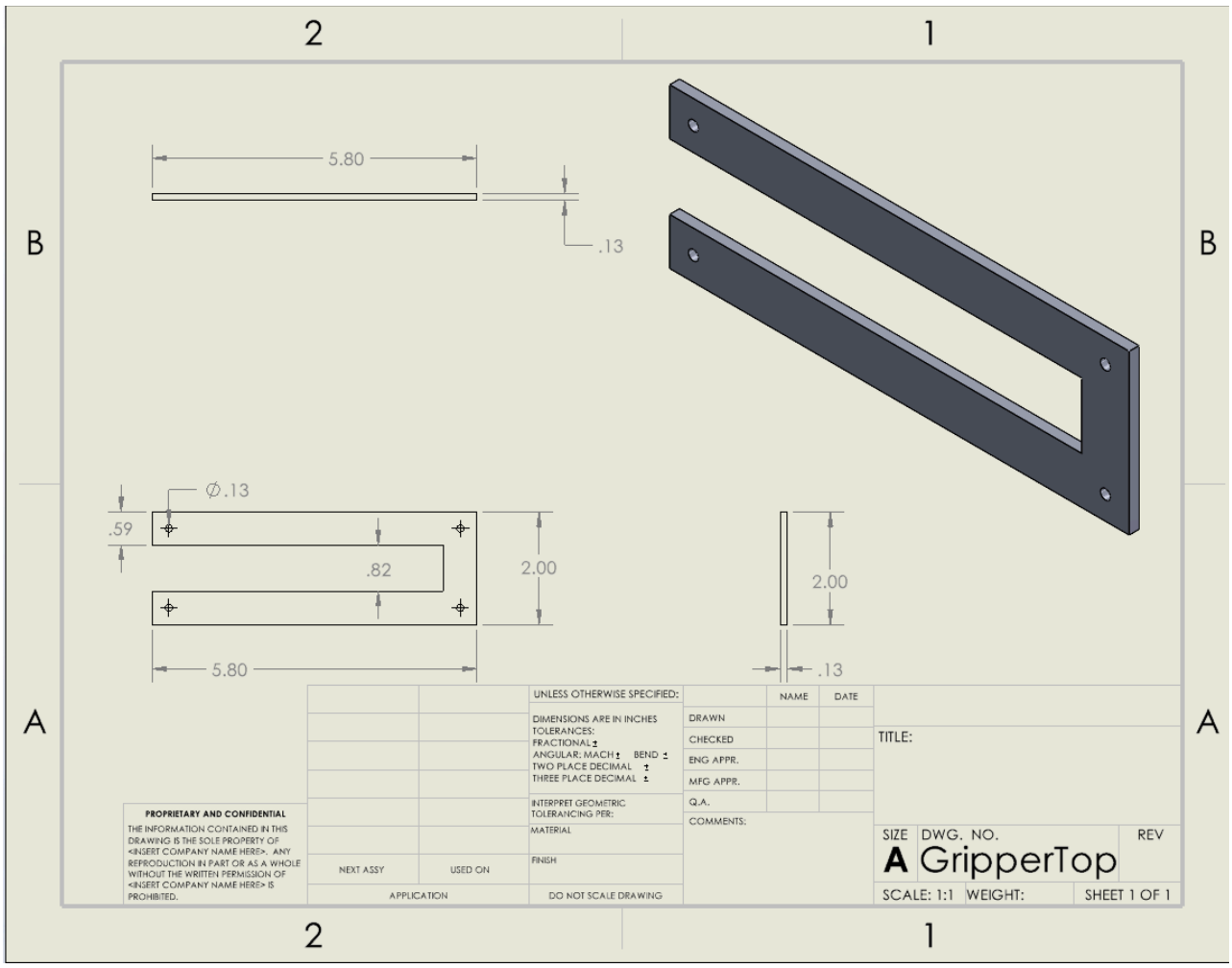
We learned many lessons as we made our way through the ENGR 7B course and the design process. One of these lessons included task assignment. Each of our team members agreed that the process of constructing their rover last quarter in ENGR 7A was stressful due to the inconsistency of task assignment and management throughout the quarter. As such, we learned to stay on task with what we were assigned from the start of the quarter and to ask for help whenever help was needed. In fact, our consistency with completing our required tasks allowed us to finish the construction of our rover on time and generated more time for testing. Additionally, we learned to prioritize initial research when beginning the project. At first, the team felt as if research was not an overly-critical portion of the project. However, as time went on, this proved to be untrue. The research we conducted in the middle of the quarter on coding mechanics to maximize the efficiency of the Arduino code was critical in allowing the team to create working code that could be tested throughout the course of the last three weeks. This research also proved vital to the usage of the PixyCam. Without prior research on the capabilities of the PixyCam, we wouldn't have been able to calibrate it correctly to find the can and allow the Arduino to close the gripper once the width of the can was wide enough for activation. These two lessons served to be very important in the process of designing,

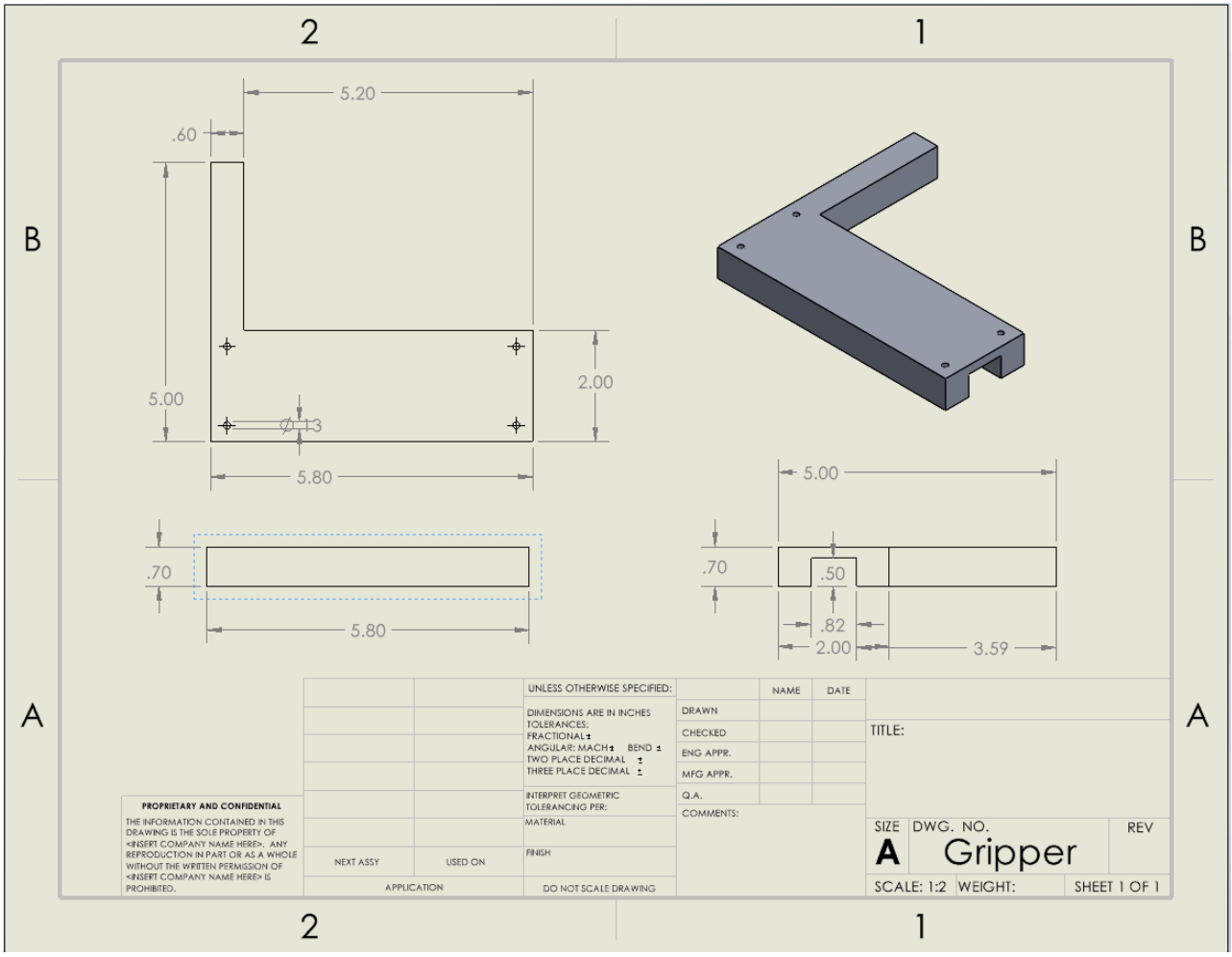
constructing, and testing a solid rover, and these lessons will be carried over past this quarter to future engineering projects.

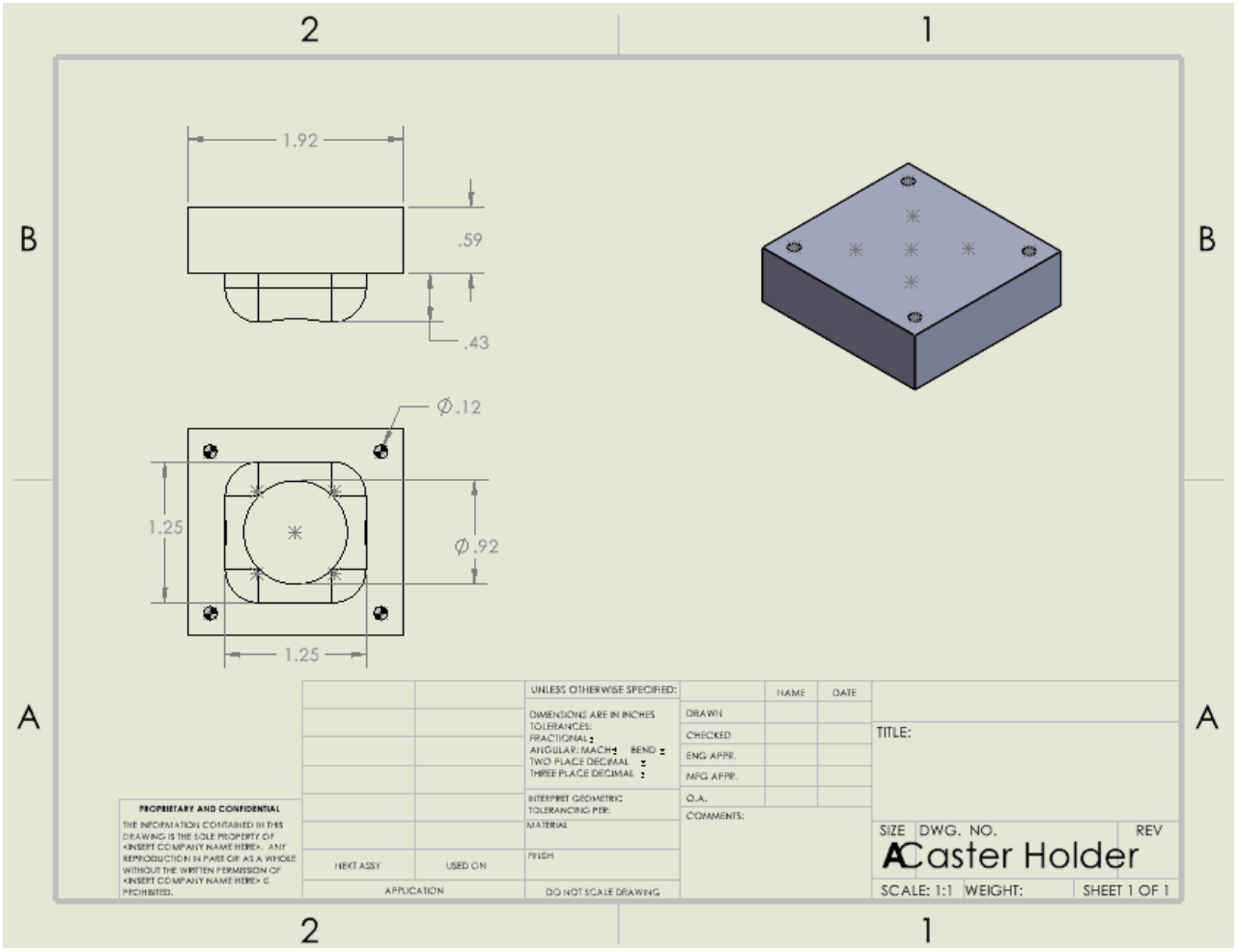
Appendix A: SolidWorks Drawings

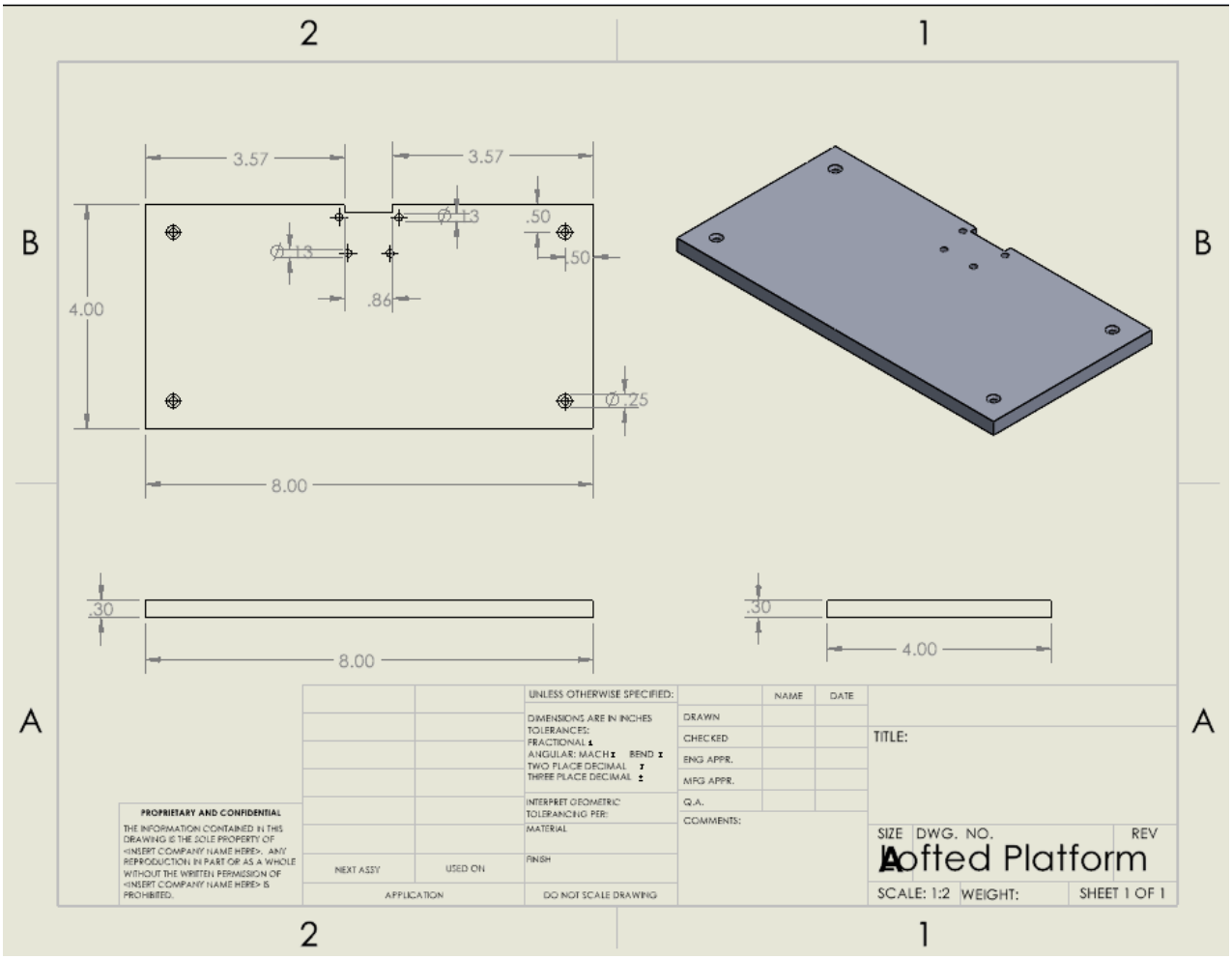


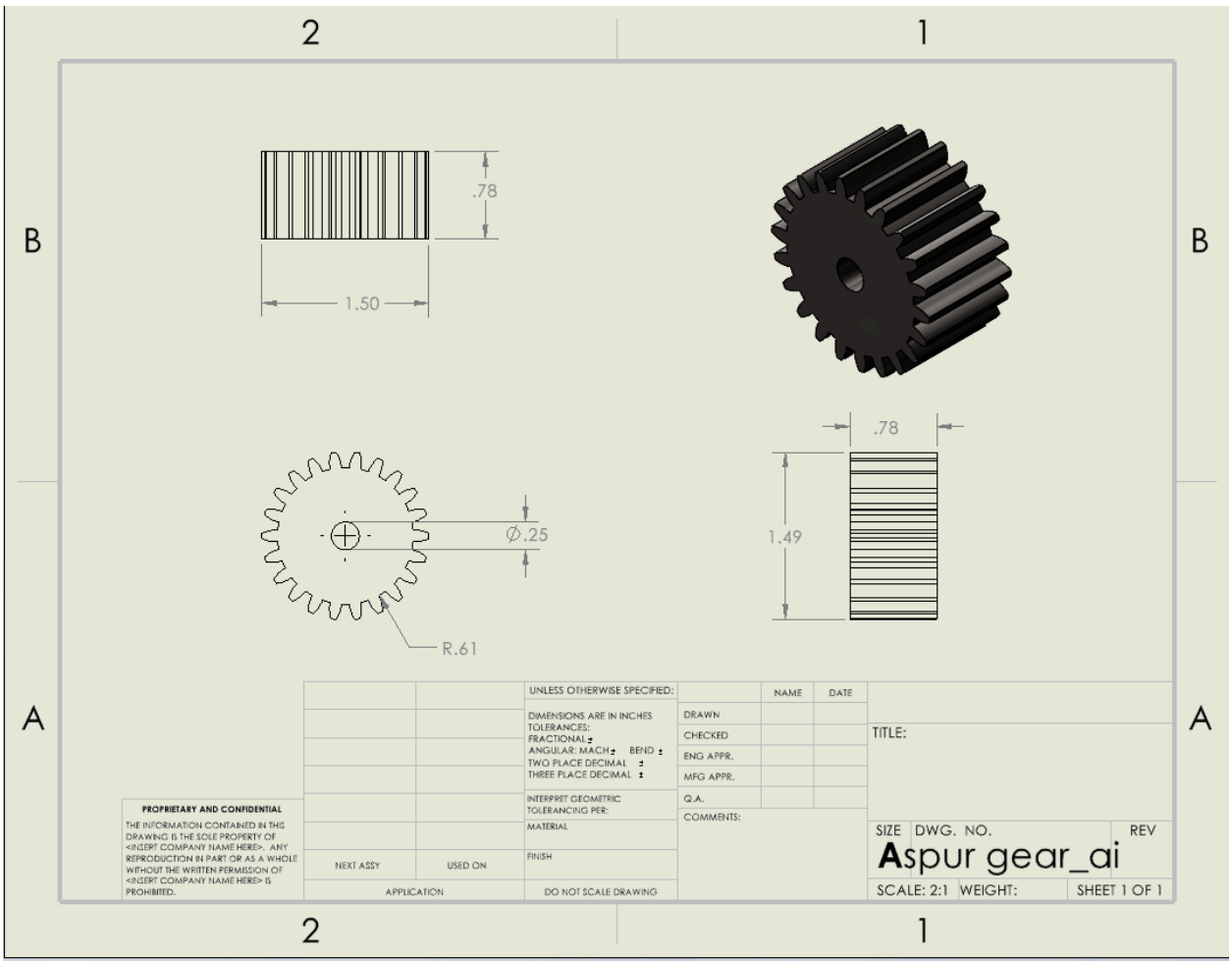

2
1
B
A
9.00
5.80
.25
.25
R1.60
R1.60
R.13
R.13
3.15
3.15
Ø.13
Ø.13
.31
Ø.13
Ø.13
5.80
9.00
.25
5.80
2.06
3.15
.25
10.42
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
A
DWG. NO.
Chassis
REV
SCALE: 1:4
WEIGHT:
SHEET 1 OF 1
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

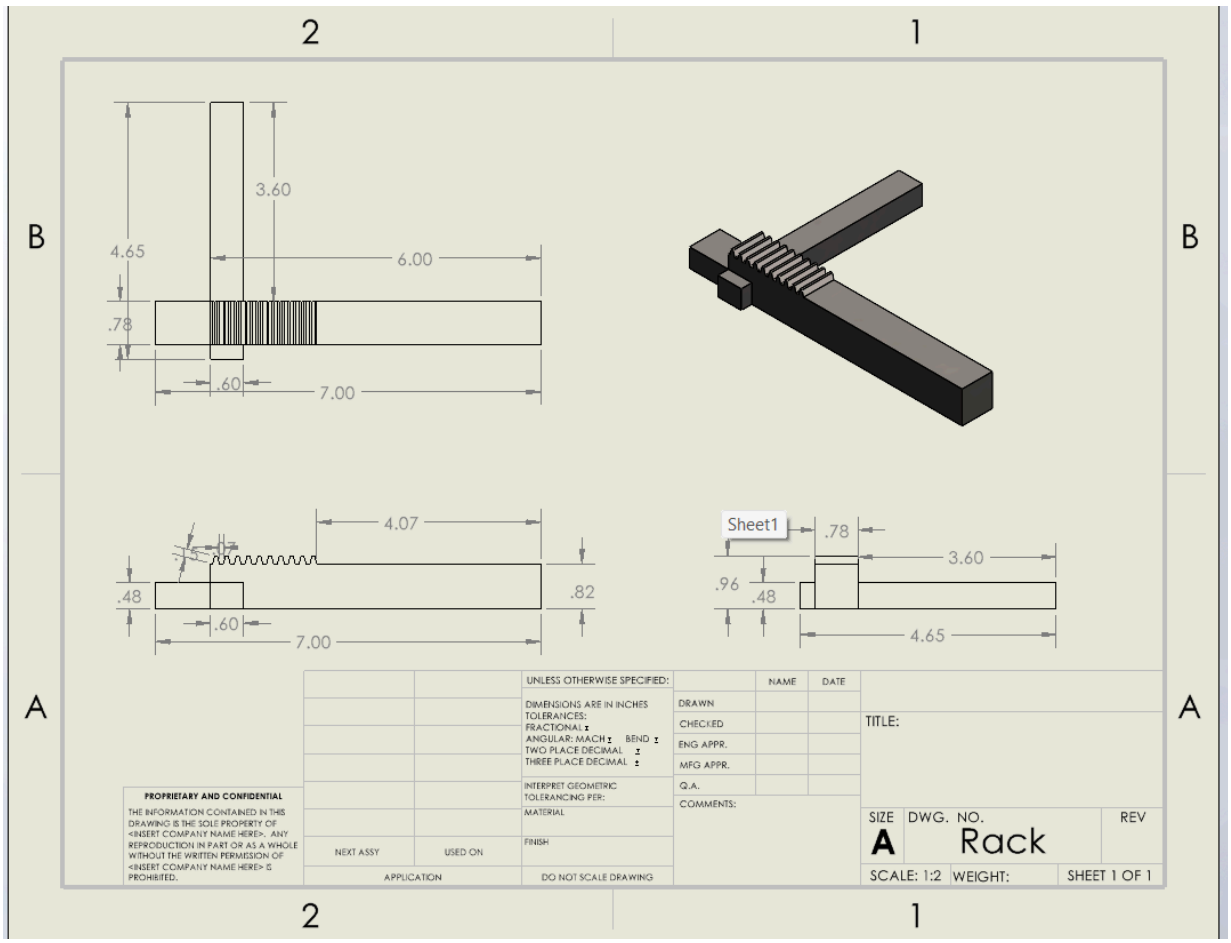

2
1
B
A
5.80
.13
Ø.13
.59
.82
2.00
5.80
2.00
.13
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
DWG. NO.
REV
A
GripperTop
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

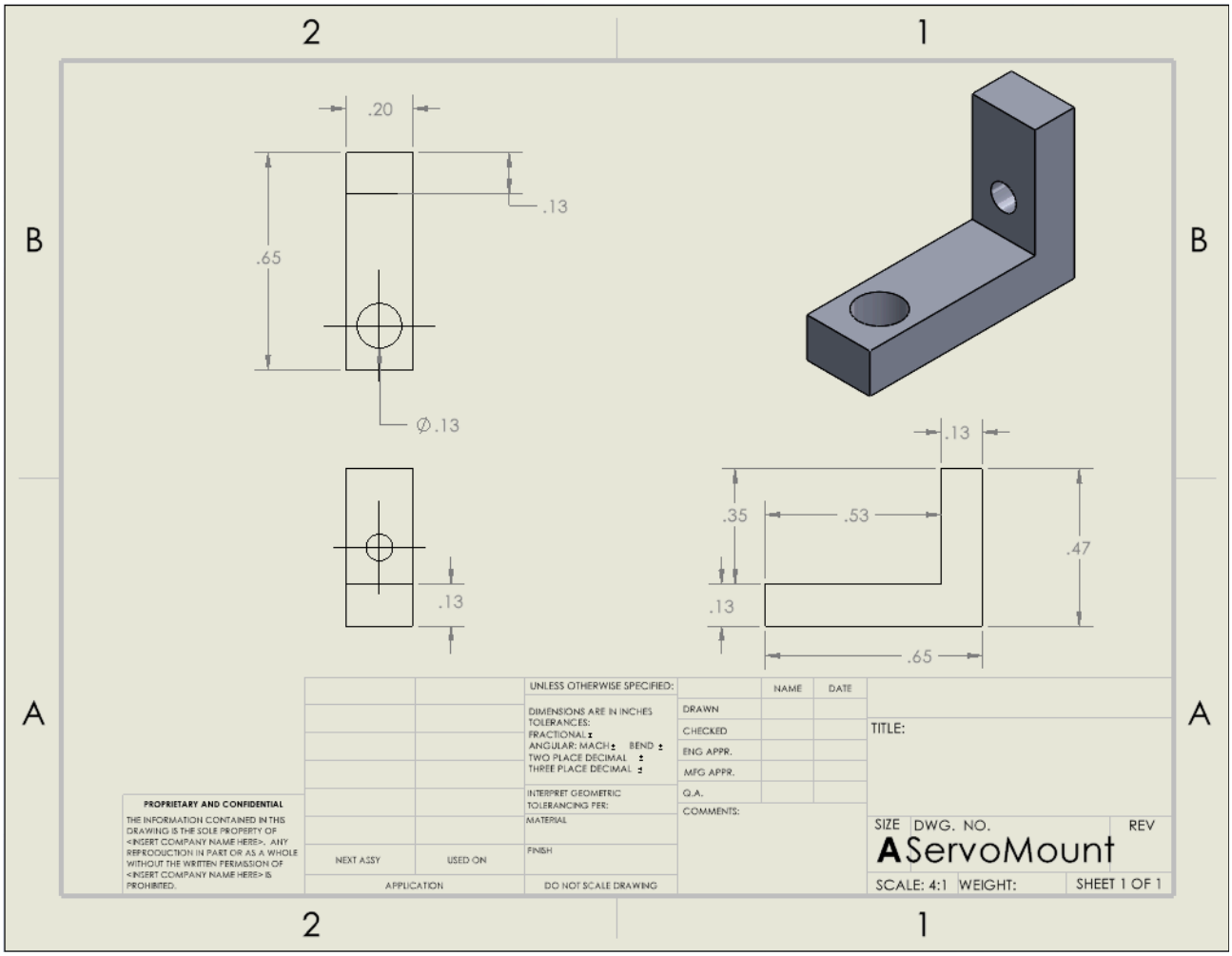

2
1
B
A
5.20
.60
5.00
2.00
Ø.13
5.80
.70
5.80
5.00
.70
.50
.82
2.00
3.59
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
A
DWG. NO.
Gripper
REV
SCALE: 1:2
WEIGHT:
SHEET 1 OF 1
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

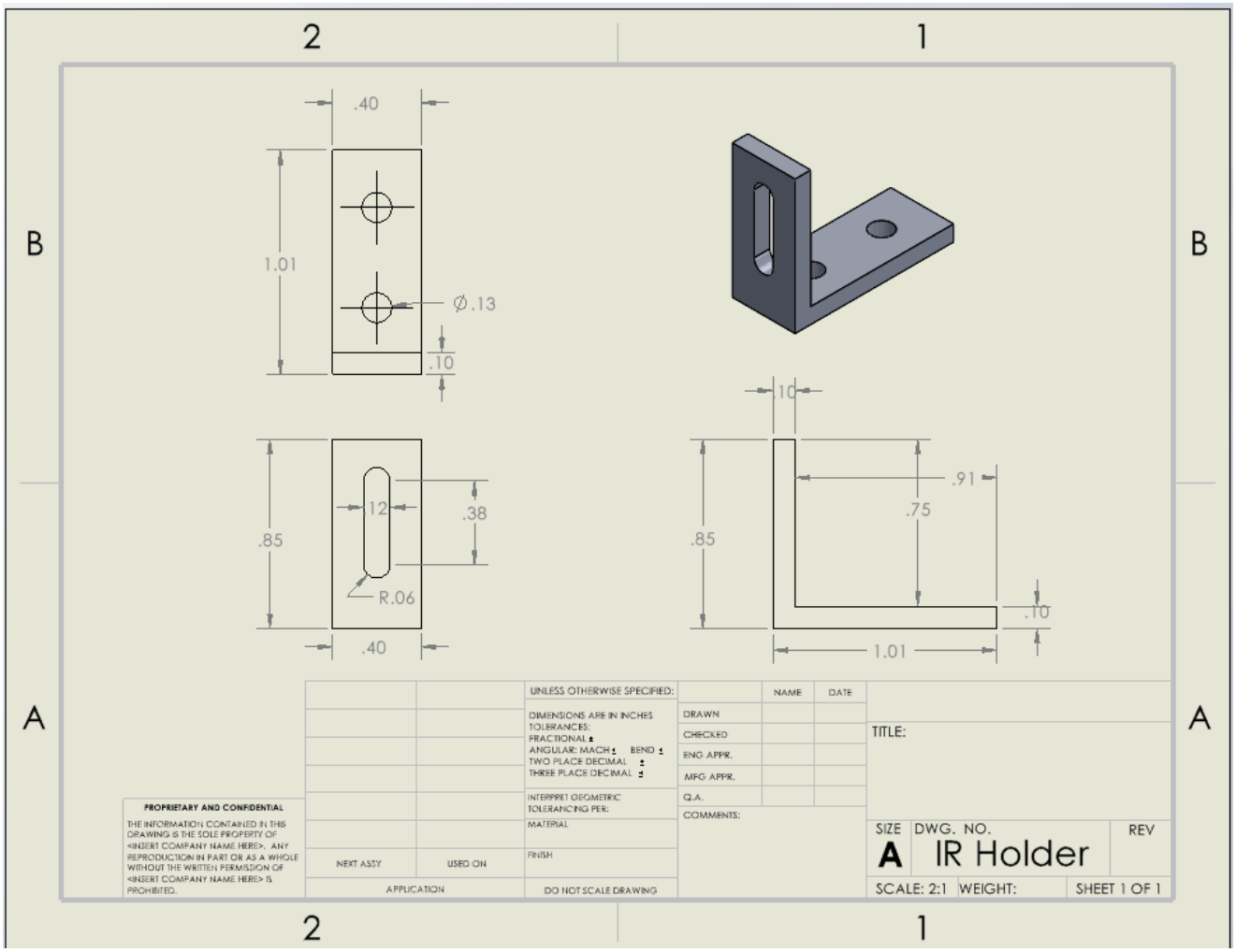

2
1
B
A
1.92
.59
.43
Ø.12
1.25
Ø.92
1.25
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL
ANGULAR: MACH BEND
TWO PLACE DECIMAL
THREE PLACE DECIMAL
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
DWG. NO.
REV
ACaster Holder
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.


2
1
B
A
3.57
3.57
Ø.13
.50
Ø.13
.50
.86
4.00
Ø.25
8.00
.30
8.00
.30
4.00
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
DWG. NO.
REV
Lofted Platform
SCALE: 1:2
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

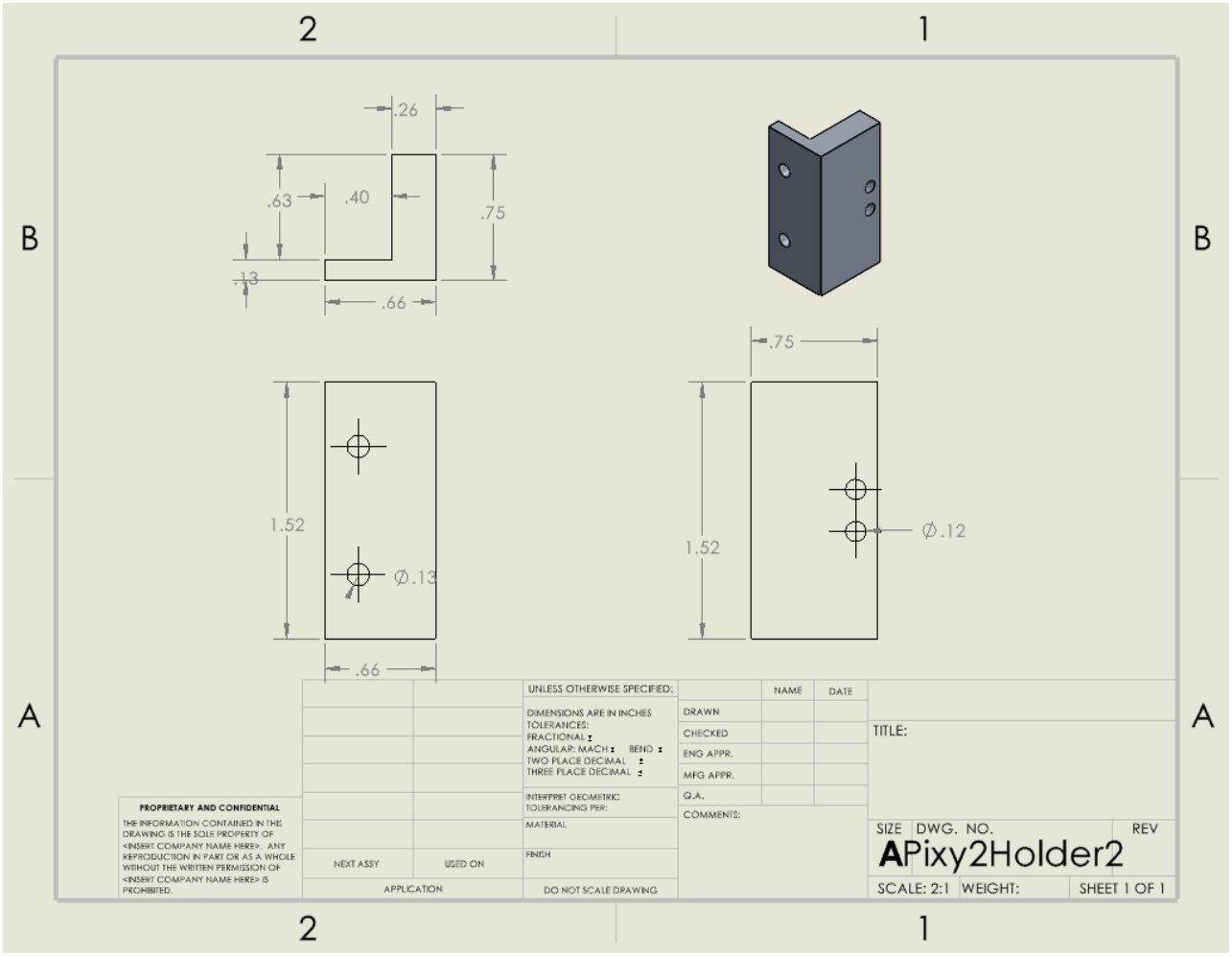

2
1
B
A
.78
1.50
.78
1.49
Ø.25
R.61
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
DWG. NO.
REV
Aspur gear_ai
SCALE: 2:1
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.


2
1
B
A
3.60
4.65
6.00
.78
.60
7.00
4.07
.48
.82
.60
7.00
Sheet1
.78
3.60
.96
.48
4.65
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
A
DWG. NO.
Rack
REV
SCALE: 1:2
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.


2
1
B
A
.20
.13
.65
Ø.13
.13
.13
.35
.53
.47
.65
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
DWG. NO.
REV
AServoMount
SCALE: 4:1
WEIGHT:
SHEET 1 OF 1
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE	
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN			TITLE:
			CHECKED			
			ENG APPR.			
			MFG APPR.			
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.			
		MATERIAL	COMMENTS:			SIZE A / DWG. NO. IR Holder / REV
NEXT ASSY	USED ON	FINISH				
APPLICATION		DO NOT SCALE DRAWING				SCALE: 2:1 / WEIGHT: / SHEET 1 OF 1



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

PROPRIETARY AND CONFIDENTIAL

THE INFORMATION CONTAINED IN THE DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

TITLE:

SIZE | DWG. NO. | REV

APixy2Holder2

SCALE: 2:1 | WEIGHT: | SHEET 1 OF 1

Appendix B: Bill of Materials

Quantity*	# of Units In package*	Company*	Item Description*	Catalog #*	Price*	Estimated Extended Price*
1	1	Arduino	Arduino Uno	A000066	$27.60	$27.60
1	1	Cytron	Motor Driver Shield (10A 7V-30V Dual DC Motor)	SHIELD-MDD10	$22.55	$22.55
1	4	Amazon	Buck Converter	B079N9BFZC	$14.99	$3.75
1	6	Amazon	BreadBoard	B07LFD4LT6	$9.99	$1.67
1	3	Amazon	Male Tamiya Connector (14awg wire)	B07VL2B5C8	$8.69	$2.90
1	10	Amazon	Power Switch (20awg wire)	B071Y7SMVQ	$7.99	$0.80
1	2	Amazon	7.2V 3600mAh Battery	B07VLKP6RJ	$34.99	$17.50
1	1	Sparkfun/adafruit/robotshop	PixyCam 2	1906	$69.95	$69.95
2	20	Amazon	IR Sensors (3.3V and 5V digital logic)	B08DR1W3BK	$69.95	$7.00
2	12	Amazon	SG90 Microservo (180 degree rotation)	B08FJ27Q1H	$20.99	$3.50
25	450	Amazon	LEDs (*Red/Green (470Ω), Yellow (220Ω), White (100Ω), Blue (33Ω)*)	B073QMYKDM	$12.99	$0.72
50	1000	Amazon	Resistors	B08FD1XVL6	$10.99	$0.55
0.5	1	Amazon	Wire (100ft 20awg)	B07SJ44SN1	$16.99	$8.50
1	1	Pololu	34:1 Gearmotor	3204	$28.95	$28.95
4	4	Amazon	65mm x 26mm Wheels	B00K73YYBQ	$12.53	$12.53
2	2	Pololu	Hex Wheel Adapter	2684	$4.95	$4.95
2	2	Pololu	3/4" Caster Wheel	955	$3.95	$3.95
2	2	Pololu	Motor Bracket	2676	$7.95	$7.95
5	1	Bamboo	3D Printing		$4.00	$20.00
1	1		1/8 polycarbonate 16"x10"		$7.22	$7.22
					Subtotal	$252.51
				*Tax rate:	7.75%	$19.57
					TOTAL ORDER PRICE	$272.08

Appendix C: Arduino Code

```
/ FINAL SAMPLE CODE

// TEAM NAME: Rhyn0
// TEAM MEMBERS: Aiden Omeed Banan, Alex Massoumi, Tim Monroe, Tobi Ogundiwin, Aras Vakilimafakheri, Michael Brian Willoughby
// DATE OF LAST EDIT: March 12, 2024



// CODE PURPOSE: This code is for a line following rover with the goal of following a black line, using IR sensors, detecting a can, with a pixy2 camera, and grabbing the
//               can with some claw attachment.

// CODE FUNCTION:The following code starts by telling the rover to follow a line and make adjustments depending on the IR sensor readings. Once the rover reaches the end of
//               of the track it will switch to utilizing the pixy2 camera to detect the can. Once it detects the can, the rover will center the object relative to the rover,
//               utlizing the pixy camera's readings. When the object is centered, the rover will continue forward until the can is in range of the claw. When the can is within
//               the claws range, the claw will close and the rover will stop moving.




// Calling Claw, Servo and Pixy Libraries
#include <CytronMotorDriver.h>
#include <Servo.h>
#include <Pixy2.h>



// Inititializing IR Pins
#define R_S 2   // IR sensor Right
#define L_S 5   // IR sensor Left
```

```
// Configure the motor driver.
CytronMD motor1(PWM_DIR, 3, 4);  // PWM 1 = Pin 3, DIR 1 = Pin 4. Left
CytronMD motor2(PWM_DIR, 6, 7); // PWM 2 = Pin 6, DIR 2 = Pin 7. Right



Servo Claw;  // Creates servo object to control claw



int pos = 180;    // Variable to store the servo position
int desired = 130; // variable to find desired width
int state = 1; // variable to define the state of the code




Pixy2 pixy; // Creates pixy objext to control pixy2 camera



// Sets up a couple things for rover

void setup()
{
 Serial.begin(9600);
  //Set IR Sensors as pins
 pinMode(R_S, INPUT);
 pinMode(L_S, INPUT);
  // Attach servo to pin 9 and set claw to open
 Claw.attach(9);
 Claw.write(pos);

 //Initialize pixy
 pixy.init();
```

```
 delay(1000);
}



// Loops through cases


// Rover starts with the line folliwng code and then switches to the claw centering code once both IR Sensors detect a line
// Then once the object is centered relative to thge claw, the rover will switch to the can retrieval segment of the code
// Making the rover go forward until the can is in reach of the claw

void loop()
{
 switch(state)
 {
   case 1:
   line_following();
   break;

   case 2:
   claw_centering();
   break;

   case 3:
   claw_retrieval();
   break;
 }

}




// Below are the 3 Case Functions:
//     1. Line Following - Code that makes the rover follow the black lines
```

```
//     2. Claw Centering - Code that centers the can in the middle of the pixy
//     3. Claw Retrieval - Code that tells the rover to go forward and grabs the can



// Function for making the rover follow the lines
   //  motor control functions - (forward, turnRight, turnLeft, stop) - are defined at bottom of file

void line_following()
{
  if (digitalRead(R_S) == 0 && digitalRead(L_S) == 0) // If neither sensor detects anything
 {
   // Rover moves forward
   forward();
   delay(3);
 }
 else if (digitalRead(R_S) == 1 && digitalRead(L_S) == 0) // If the right sensor detects the line
 {
   // Rover turns right
   turnRight();
   delay(15);
 }
 else if (digitalRead(R_S) == 0 && digitalRead(L_S) == 1) // If the left sensor detects the line
 {
   // Rover turns left
   turnLeft();
   delay(15);
 }
 else if (digitalRead(R_S) == 1 && digitalRead(L_S) == 1) // If both sensors detect the line
 {
   // Rover stops moving
   stop();
```

```
   // Rover switches states and begins centering the can
   state=2;
 }
}



// Function for Centering the Can relative to the Rover Claw

void claw_centering()
{
 // Print information about can location
 pixy.ccc.getBlocks();
 int object_x = pixy.ccc.blocks[0].m_x;
 int object_width = pixy.ccc.blocks[0].m_width;
 Serial.println(object_x);



  if (object_x < 185) // If can is to the right of the claw
  {
   // Turn Right
    motor1.setSpeed(64);
    motor2.setSpeed(-128);
  }
  else if (object_x > 195) // If can is to the left of the claw
  {
   // Turn Left
    motor1.setSpeed(-128);
    motor2.setSpeed(64);
  }
 else // If can is within claw width
  {
   // Rover stops
   motor1.setSpeed(0);
   motor2.setSpeed(0);

   // Switch case to claw retrieval function
   state=3;
```

```
  }
 delay(100);
}



// Function for moving towards the can and closing the claw

void claw_retrieval()
{
 // Print information about the can location
 pixy.ccc.getBlocks();
 int object_x = pixy.ccc.blocks[0].m_x;
 int object_width = pixy.ccc.blocks[0].m_width;
 Serial.println(object_x);

   if (object_width < desired) // If can is outside claw range
   {
     // Rover goes forward
     motor1.setSpeed(-250);
     motor2.setSpeed(-250);
   }

   else if (object_width >= desired) // If can is in range of claw
   {
     // Claw closes
      Claw.write(80);

      // Rover stops
      motor1.setSpeed(0);
      motor2.setSpeed(0);
   }
 delay(10);
}
```

```
//Motor Control Functions - (For the line following code)

void forward() // Makes Rover go forward
{
 motor1.setSpeed(-150);   // Motor 1 continues forward at 50% speed.
 motor2.setSpeed(-150);   // Motor 2 continues forward at 50% speed.
}


void turnRight() // Makes Rover turn Right
{
 motor1.setSpeed(256);   // Motor 1 (Left) runs forward at 50% speed.
 motor2.setSpeed(-180);   // Motor 2 (Right) set at -25% speed.
}


void turnLeft() // Makes Rover turn left
{
 motor1.setSpeed(-180);   // Motor 1 (Left) set at -25% speed.
 motor2.setSpeed(256);   // Motor 2 (Right) runs forward at 50% speed.
}


void stop()  // Stops Rover
{
 motor1.setSpeed(0);   // Motor 1 (Left) stops.
 motor2.setSpeed(0);   // Motor 2 (Right) stops.
}
```

Appendix D: References

Amazon.Com, www.amazon.com/ref=as_li_ss_tl?&hvadid=160386164570&hvpos=1t1&hvnetw=g&hvrand=12825922147427444940&hvpone=&hvptwo=&hvqmt=e&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9016899&hvtargid=kwd-10573980&ref=pd_sl_7j18redljs_e&linkCode=sl2&tag=theascensions-20&linkId=f6f27dac1ce0817909c8e4aa678f00f2. Accessed 22 Mar. 2024.

Cytron Technologies, www.cytron.io/. Accessed 21 Mar. 2024.

"Mars Exploration Rovers." *NASA*, NASA, 16 Jan. 2024, mars.nasa.gov/mars-exploration/missions/mars-exploration-rovers/.

"Pixy2 Downloads." *PixyCam*, PixyCam.com/downloads-pixy2/. Accessed 21 Mar. 2024.

"Pololu Robotics and Electronics." *Pololu Robotics & Electronics*, www.pololu.com/. Accessed 21 Mar. 2024.

MAE 3 - Gear ratios. (n.d.). https://mae3.eng.ucsd.edu/machine-design/gear-ratios Accessed 21 Mar. 2024.