

Operations Manual

Team #11

Robo-Weeder

4/1/2016

Submitted to: Dr. Gupta (Faculty Advisor)
Jeff Phipps (Sponsor)

Authors: Steven Miller (ME)
Christopher Murphy (ME)
Arriana Nwodu (ME)
Aquiles Ciron (EE)
Steven Williamson (EE)
Zhang Xiang (ME)

Table of Contents

Table of Figures.....	i
Table of Tables	ii
Abstract	iii
Acknowledgements	iv
1. Introduction	1
2. Functional Analysis.....	1
3. Project/Product Specification	1
4. Product Assembly	3
5. Operation Instructions	5
6. Troubleshooting.....	7
7. Maintenance.....	8
8. Conclusion.....	10
Appendix A	11
Appendix B.....	17
Appendix C	18

Table of Figures

#	Description	Pg.
1	Auger Removal	5
2	Transmitter Functions	6

Table of Tables

#	Description	Pg.
1	Motor Parameters for Robo-Weeder	2
2	Roboclaw Motor Controllers	3
3	Spare Parts Inventory	9

Abstract

The Robo-Weeder senior design project is sponsored by Jeff Phipps M.E., the owner of Orchard Pond Organics. Orchard Pond Organics is an 8 acre organic farm that has a pressing need for the assistance in weed control. The primary objective of the Robo-Weeder senior design project is to design and create a robotic system that will aid in this need by using a shearing mechanism to remove weeds in between the rows of planted vegetables. Team 11 has contacted and met with Mr. Jeff Phipps, the project sponsor, and discussed the final design. The final Robo-Weeder design has been approved by the sponsor as well as the faculty advisor and the final Robo-Weeder design has been built and tested by Team 11.

Acknowledgements

Team 11 would first like to thank our Faculty Advisors Professors Nikhil Gupta and Jerris Hooker. They've been invaluable in the development of the Robo-Weeder. They've guided us not only on the technical aspects, but also in team morale and because of that we learned to perform in an optimal and cohesive manner. We would also like to thank our team's sponsor Mr. Jeff Phipps of Orchard Pond Organics. Due to his vision of providing more nutritious food at a lower cost, thousands if not millions of low income families can have the option of a healthier diet. Finally we would like to thank the FAMU-FSU College of Engineering Machine shop as well as Brandon Shaw of the Applied Superconductivity center for aiding in the assembly process.

1. Introduction

The Robo-Weeder senior design project was created with the desire to improve the production rate of organic farms. The project consists of the design and fabrication of a robotic platform that will assist in weed control on the crop seedbed. The use of technology on farms is not a new idea, but it is becoming increasingly more popular.

Organic farming describes a method of farming that moves away from the traditional farming techniques of the mass application of herbicides and pesticides to control unwanted pests and weeds. Organic farming also steers away from tilling of the soil to control the growth of unwanted weeds. Leaving the soil undisturbed promotes biodiversity and improves its quality by allowing microorganisms such as earthworms to naturally aerate the soil. With the absence of chemicals and tilling methods to control pests and weeds, farmers are confronted with a massively labor intensive task that requires the manual removal of weeds from the seedbed.

The Robo-Weeder robotic system will face many challenges that will include the method of weed control, the power system as well as how the operator controls the robot. These challenges and many others will be addressed in this paper. The team assigned to the Robo-Weeder senior design project consists of four mechanical engineers and two electrical engineers and is sponsored by the FAMU-FSU Mechanical Engineering Department. The project's sponsor is Mr. Jeff Phipps of Orchard Pond Organics and the advisors from the college of engineering include Dr. Nikhil Gupta and Dr. Jerris Hooker.

2. Functional Analysis

The Robo-Weeder robotic system is a “Proof of Concept” prototype designed to be a solution to the massive labor needs that are present on organic farms. The Robo-Weeder is designed to reduce the total manpower needed by assisting in the removal of unwanted weeds from the seed bed of the crops. The Robo-Weeder is designed with a auger style shearing device that when in contact with the ground applies a shear force to the surface of the seedbed. This shear force is then translated to the weed where it is uprooted and displaced on the seedbed. For peak performance, the Robo-Weeder should be deployed onto newly formed seedbeds and be used as a preventive tool against young, freshly sprouted weeds due to the vulnerable nature of the weeds during this stage of growth.

The operator for the Robo-Weeder will need to be in close proximity of the robot to ensure proper adjustment of the speed of the augers, their level of contact with the surface as well as the direction of travel of the robot. All user control is achieved through the use of a wireless transmitter similar to that used by RC (Radio Controlled) Hobbyists.




3. Project/Product Specification

When considering the primary function of the Robo-Weeder project, the most important subsystem to consider would be the weeding mechanism. The Robo-Weeder’s weeding

mechanism is centered on two augers which are 8 inches in length, 4 inches in diameter with a 1.3” pitch. The size of the augers is the most important as it is the working zone of the robot and provides the weeding function. The 4 inch diameter of the auger is important due to the torque requirement needed to rotate the auger with an applied load. At the most extreme case, the augers will penetrate a maximum of 1 inch into the soil. The length of the augers is also important as the length dictates the physical size of the lane that could be weeded during a single pass. With a length of 8 inches for each auger, the auger pair share a 1 inch offset which in turn widens the effective weeding lane to 9 inches and also works to prevent an erosion of soil. The erosion of soil is countered by the opposing flighting found on each auger; the front is left handed while the rear is right handed. The final auger assembly is located inside a housing that allows for the connection of motors as well as the positional control. The final weeding mechanism housing that is connected to the chassis of the Robo-Weeder measures to be 13.5 inches wide by 14 inches long.

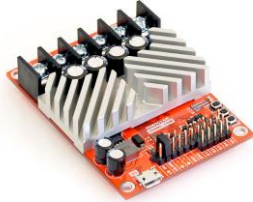
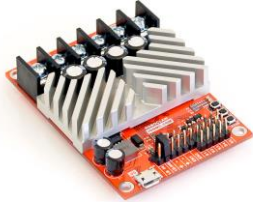
The motors used to operate the Robo-Weeder vary in shape and size depending on the application. **Table 1** below outlines the key parameters and functionality for each motor housed on the Robo-Weeder.

Table 1: Motor Parameters for Robo-Weeder

<p>PG188 Gearmotor (Steering Motor)</p> 	<p>PG71 Gearmotor (Drive Motor)</p> 	<p>RS540 Motor w/ 104:1 Gear (Shearing Motor)</p> 
No-Load RPM: 28	No-Load RPM: 75	No-Load RPM: 162
Stall Torque: 396 in-lbs.	Stall Torque: 200 in-lbs.	Stall Torque: 256 in-lbs.
Stall Current: 22A	Stall Current: 22A	Stall Current: 42A

The electronic system housed on the Robo-Weeder is diverse in nature and allows for the control of up to 6 DC motors. To control these 6 DC motors, three Roboclaw motor controllers are implemented. Each Roboclaw motor controller is able to control up to 2 motors each: 2 for the drive function, 2 for the steering function, and 2 for the shearing function. Two different types of Roboclaw motor controllers are used: the 2x30A and 2x45A. The specifications for each motor controller are shown below in **Table 2**.

Table 2: Roboclaw Motor Controllers

<p style="text-align: center;"><u>Roboclaw 2x30A</u></p> 	<p style="text-align: center;"><u>Roboclaw 2x45A</u></p> 
<p style="text-align: center;">Motor Channels: 2</p>	<p style="text-align: center;">Motor Channels: 2</p>
<p style="text-align: center;">Minimum Operating Voltage: 6V</p>	<p style="text-align: center;">Minimum Operating Voltage: 6V</p>
<p style="text-align: center;">Maximum Operating Voltage: 34V</p>	<p style="text-align: center;">Maximum Operating Voltage: 34V</p>
<p style="text-align: center;">Continuous Operating Current: 30A/ch.</p>	<p style="text-align: center;">Continuous Operating Current: 45A/ch.</p>

Two of the Roboclaw 2x45A will be used in order to control the drive and shearing mechanism of the Robo-Weeder while the Roboclaw 2x30A will be used to control the 2 steering motors. The entire Robo-Weeder design is to be operated on a 12V battery system and will draw approximately 55A of current through normal operation.

4. Product Assembly

The Robo-Weeder can be broken down into 4 main subassemblies: Frame, Steering and Drive, Auger Housing and Lift. Each assembly needs to be assembled individually and then mounted to the central frame.

3-D Models

For simplification, the 3-D model and exploded view for each of the subassemblies can be found in Appendix A.

Tools

- Common and Phillips Screwdriver
- Socket Set
- Allen Key Set
- Adjustable Wrench
- Channel Lock or Vice Grip Pliers

Frame Assembly

The frame of the Robo-Weeder is made of Aluminum square tubing which will be welded prior to assembly. Also, the vertical portion of the lift system will be welded to the central portion

of the frame. This assembly will serve as the central component of the chassis and will have all other subassemblies attached. The only preparation of the frame is the insertion of four ½ inch bushings into the front and rear drive steering assembly mount locations.

Drive and Steering Assembly

The drive and steering assembly is a singular assembly but there will be two of them. One located in the front and the other in the rear. For the assembly, the socket set, Allen key set, and pliers will be needed. Firstly, attach the steering pin to the main wheel housing by passing the pin up through the inside and use four bolts. Next, attach one flange bearing to the outer edge of each side of the main wheel housing using two bolts each. With the bearings now attached, insert the axle shaft through each bearing, making sure to mount the 18 tooth sprocket inside the wheel housing. Next secure the axle's position by pressing two e-clips into the two small grooves located on the inside of the wheel housing and securing the set screws onto the axle located on the flange bearings. With the axle and sprocket installed, insert the ¼ inch shaft bushing in the upper hole on the wheel housing. Proceed to mount the drive motor to the drive motor plate using 4 M4 screws, and install the shaft motor shaft extension with the 10 tooth sprocket. Now insert the drive motor extension into the ¼ inch bushing and use three 5-32 screws to mount the motor to the wheel housing. Ensuring that the sprockets are aligned, tighten the set screws on the sprockets to lock the sprockets to their prospective shaft. Finally, mount each 10 inch diameter pneumatic wheel on either side of the wheel housing and secure the wheels with a locking nut. Also, prepare the steering motor for installation by using 4 M4 screws to attach the motor to the steering motor mount and installing the coupler onto the steering motor shaft. Repeat process for the second drive and steering assembly.

Auger Housing Assembly

For assembly of the Auger Housing prepare the socket set, Allen keys and pliers. It should be noted that the auger housing is designed to allow removal of the augers. Locate the front, back, left, right, motor mount and top plates that make the main housing structure. Attach four flange bearings to the left and right housing plates, two each. Attach the right plate to the front, top and rear plates using ten 5-32 screws. At this time, insert the extension pins into their perspective auger and insert the extended augers into the bearings installed on the right plate. Note: Ensure that the auger with right handed flighting is installed in the front of the auger housing. With the augers inserted into the right housing plate, insert the auger extensions into the left housing plate and attach using ten 5-32 screws. Align the augers and tighten the flange bearing set screws. With the augers installed and the main housing assembled, attach the auger motors to the auger motor mount plate. Attach 10 tooth sprockets to both the motor shafts and the auger motor extension pins and mount the auger motor mount to the auger housing top plate. To remove the augers, remove 10 screws holding the back plate of the auger housing on and the set screws from the flange bearings, then remove the back plate, and now the auger may be removed. See **Figure 2** for more instructions on removal of the augers.

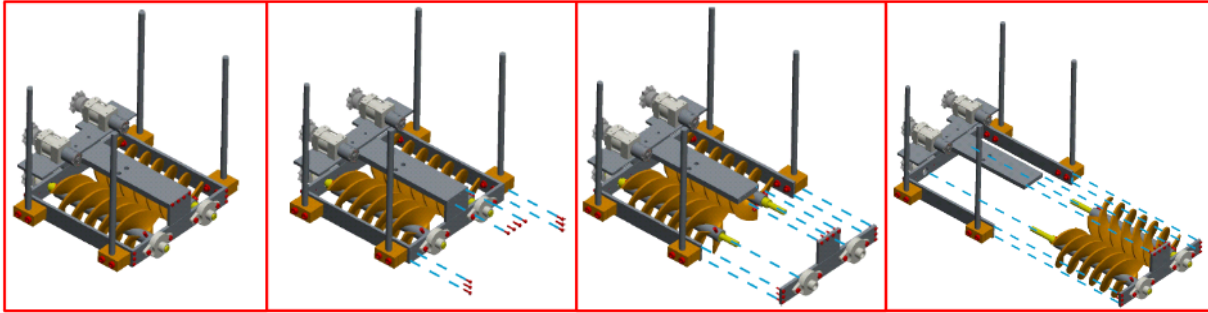


Figure 1: Auger Removal

Lift Assembly

The lift system is a relatively simple structure that is mounted directly to the auger housing and the frame. The socket set is the only tools needed for assembly. Locate the four ½ inch diameter stainless rod and the four 1 inch steel tubes. These are the stabilizing posts of the lift system and serve to keep alignment. Loosely install the inner and outer aluminum brackets to the auger housing front and rear plates using eight 5/16 inch bolts and locking hardware. Insert the ½ stainless steel rods between the aluminum brackets ensuring the rod is flush with the bottom of the bracketry and tighten the bolts. With the rods attached to the auger housing, align the 1 inch steel tubes flush with the top of their perspective mounts, then install ½ inch bushings inside the steel tubing. Using eight 5/16 bolts and locking hardware, install the tubes onto the central frame such that the tubes protrude downward. Finally, install bracketry on the vertical portion of the lift and the auger top plate for installation of the linear actuator.

Final Assembly

With each of the sub-assemblies completed, lift the frame over the auger housing and slide the ½ inch rods through the bushings in the 1 inch steel tubing. Lift the auger housing to the upper most position and install the linear actuator using the provided pins. Next, insert the steering pin of each drive and steering assembly through the bushings located at the front and rear of the frame. Secure each assembly into their prospective location by installing two e-clips into the small grooves on the steering pin just above the bushing. Locating the steering motor, insert the mounted coupler over the steering pin and mount the motors to the frame using four 5/16 bolts and locking hardware. Lock the coupler to each perspective shaft by securing the set screws located on each coupler. Also, install the drive chain for each of the drive and auger motors. See Appendix A for more information.

5. Operation Instructions

The Robo-Weeder is designed to be controlled through wireless communication using a transmitter and receiver. The transmitter used for this is the FlySky FS-T6. In order to drive the

Robo-Weeder, the y-axis of the right joystick of the transmitter will control both drive motors transmitting a signal telling the machine to drive forward or reverse. The x-axis of the right joystick will operate the front steering of the Robo-Weeder while the x-axis of the left joystick will operate the rear steering of the Robo-Weeder, allowing for the capability of parallel steering. To power on the augers, the knob on the top right of the transmitter can be twisted in order to speed up the augers from neutral to maximum speed. Once the desired auger speed is set, the augers will continue to spin at a constant RPM until the knob is twisted again. The knob on the top left of the transmitter will operate the linear actuator, lifting or lowering the auger housing bringing it closer or farther from the ground. **Figure 2** below gives a schematic view of the transmitter and its functions.

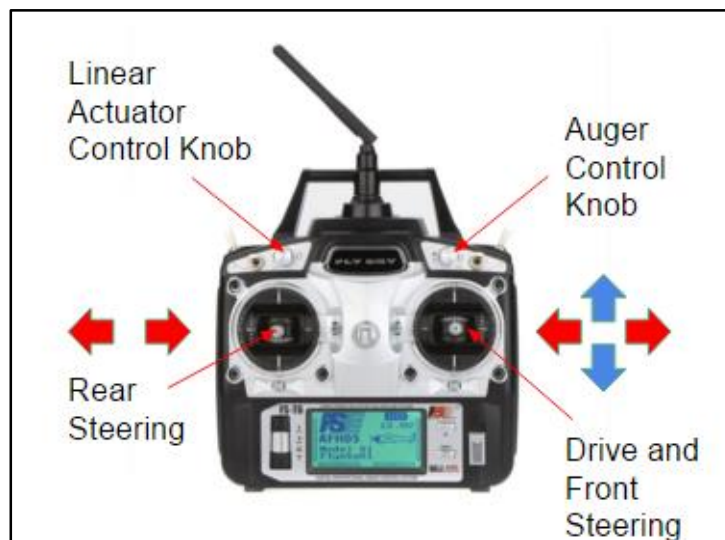


Figure 2: Transmitter Functions

With the transmitter in hand, the operator must first rotate the main power switch to the on position signified by a green window on the side of the power switch. The power switch is located on the side of the lift support tower on the robot. With the main power switch in the on position, the Robo-Weeder is now activated and ready for use. At this time the operator should move the power switch on the transmitter to the on position which will activate the transmitter allowing control of the robot. As seen in figure 2 above, the operator can now operate the Robo-Weeder.

With the robot in the correct position on the seedbed, rotate the top left knob to lower the linear actuator into position. The augers should be in loose contact with the soils surface to begin. With the augers in position, set the auger speed by rotating the top right knob until the desired speed is reached. Begin the weeding operation by using the joysticks to drive the robot down the seedbed next to the crop. Adjustment of the auger system can be made at any time during operation to achieve the correct amount of ground contact and angular velocity. When the weeding operation is complete, return both knobs to the off position to lift and stop the rotation of the augers.

6. Troubleshooting

Trouble shooting is a valuable part of the design process and allows for accurate repair and ultimate improvement of Robo-Weeder's components. In this section, the possible problems, methods of detection and their solutions will be outlined.

NOTE: Power should be disabled from the Robo-Weeder before engaging in any troubleshooting actions!

Problem: *Failure to move when drive motor is engaged.*

- **Action:** Check the locking hardware that secures the wheel to the axle shaft and ensure it is tight.
 - **Repair:** Tighten locking hardware that holds tire onto the axle.
- **Action:** Check the roller chain system to ensure chain is intact and properly aligned on sprockets.
 - **Repair:** Remove master link on roller chain, realign chain on the sprockets and reinstall link.
- **Action:** Check the roller chain system to ensure sprockets are secured to shaft.
 - **Repair:** Using an Allen key, tighten the setscrews onto the axle shaft.
- **Action:** Check the battery voltage.
 - **Repair:** Charge the batteries and ensure 12V.

Problem: *Failure to turn when steering motor is engaged.*

- **Action:** Check to ensure steering pin coupler is secured to both motor and steering shaft.
 - **Repair:** Tighten the setscrews onto both the steering motor shaft and the steering pin.
- **Action:** Check the battery voltage.
 - **Repair:** Charge the batteries and ensure 12V.

Problem: *Augers fail to turn on when auger motors are activated.*

- **Action:** Check the roller chain system to ensure chain is intact and properly aligned on sprockets.
 - **Repair:** Remove master link on roller chain, realign chain on the sprockets and reinstall link.
- **Action:** Check the roller chain system to ensure sprockets are secured to shaft.
 - **Repair:** Using an Allen key, tighten the setscrews onto the axle shaft.
- **Action:** Check to ensure augers are free of debris.
 - **Repair:** Remove debris from the auger.
- **Action:** Check the battery voltage.
 - **Repair:** Charge the batteries and ensure 12V.

7. Maintenance

Regular maintenance and inspections should be conducted on the Robo-Weeder in order to maintain optimal performance during operation. This section will outline the critical components that need to be maintained.

Auger Housing

The auger housing has the largest amount of cap screws which can become loose after prolonged use. It is recommended that each week, a full inspection of all screws in the auger housing be checked to ensure they are properly tightened. The housing is also the location of the lift rods and their subsequent mounts. It is recommended that the bolts associated with the attachment of the lift rods be checked weekly. If it is determined that any screw or bolt is loose, tighten immediately before using the Robo-Weeder.

Another key component of the auger housing is the flange bearings which are secured to the auger shafts. It is recommended that after every use that these bearing are inspected to ensure sand and debris has not become lodged around the bearing. If debris of any kind is present, remove to insure premature failing of the bearings does not occur. The roller chain and sprockets associated with each auger should also be checked to insure proper tightness and alignment. Oil should also be added to the roller chains monthly to ensure proper lubrication.

Steering and Drive

The steering and drive system on the Robo-Weeder is fairly robust and only minor maintenance will need to be performed to maintain proper. As mentioned with the Auger housing, inspect both the flange bearings connected to the axle shaft as well as the roller chain and sprocket system. Ensure proper lubrication and tension of the roller chains and check that the bearings are free of dirt and debris.

Electrical System

Regular maintenance of the electrical system comes in the form of ensuring that all the wires are in good condition, no frayed wires or damages in the distribution box. The plastic housing for the electrical system has to be cleaned regularly and ensure that proper seal is intact, ensuring protection from the element. The battery itself has to be kept in a constant state of charge, never leave the battery in a state of discharge for an extended period of time. The terminal connections to the motors, batteries, motor controllers have to be inspected regularly after each use of the Robo-Weeder, ensuring that no wires are loose from its connection. The transmitter has to be kept in a secure place as well as kept clean at all times.

Routine maintenance of the electrical system needs to be completed after each use of the Robo-Weeder. Always keep the electrical system clean at all times. The key feature to be kept clean constantly are the motor encoders that are built-in within the motor component itself, always make sure that the encoder is kept clean at all times, no visible damage to the wires or the encoder itself.

Spare parts

Spare parts were necessary for design, building, and test of the Robo-Weeder's individual subassemblies, as well as the testing of the Robo-Weeder as a whole. Below in **Table 3** spare parts are listed in by either electrical or mechanical based. They are also categorized by item, unit cost, quantity, total cost, and supplier.

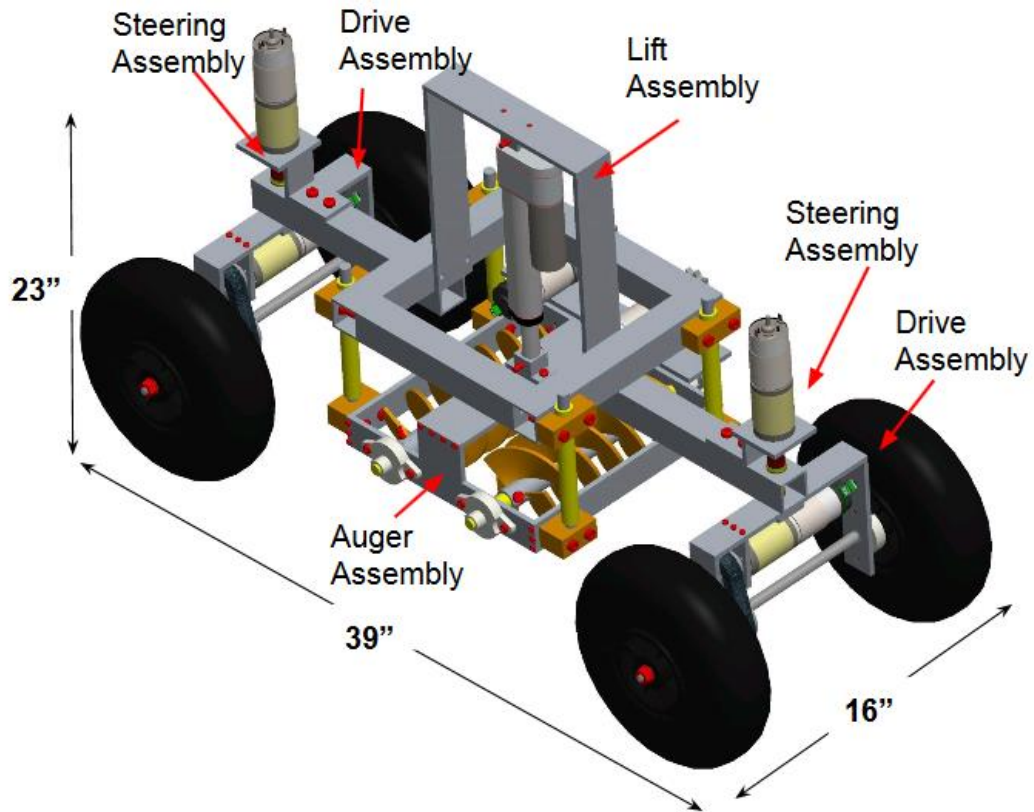
Table 3: Spare Parts Inventory

Component	Item	Unit Cost	Quantity	Total Cost	Supplier
Electrical	Radiolink Transmitter	\$36.81	1	\$36.81	Robot Shop
	Arduino Mega	\$34.99	1	\$34.99	Robot Shop
	12V Power Supply	\$23.97	1	\$23.97	Amazon
	Wiring	Varies	Varies	Varies	DigiKey
Mechanical	Nuts and Bolts	Varies	varies	Varies	McMaster

8. Conclusion

The Robo-Weeder senior design projects primary goal was to design and fabricate a remotely operated robotic system to aid in weed control on organic farms. The weeding mechanism that is housed on the Robo-Weeder is an auger design that will remove weeds by applying a shear force to the soils surface resulting in the uprooting of the weeds. The Robo-Weeder's many functions are controlled by six DC gear motors. The central unit controlling the electronics located on the RObo-Weeder is an Arduino Mega microcontroller which in turn sends signal to three Roboclaw motor controllers. The Robo-Weeder is designed to be operated by a user that is in close proximity to the robot allowing for quick corrections in weeding mechanism. With the ability to manipulate auger speed and ground contact, the Robo-Weeder will greatly reduce the physical labor required to control weed growth on organic farms.

Appendix A



Full assembly of Robo-Weeder with subassembly labels.

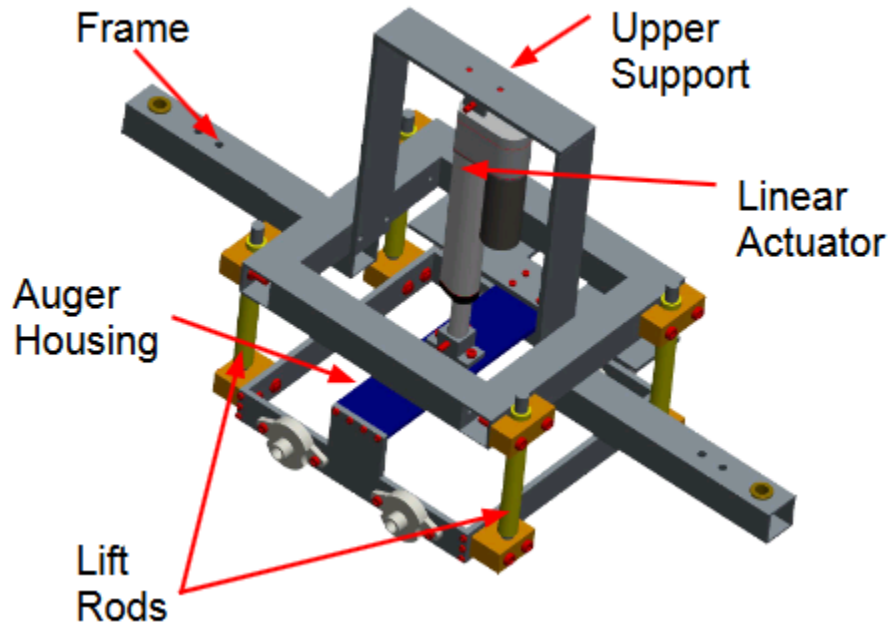


Figure 3: Frame and lift subassembly connected to auger housing, with labels.

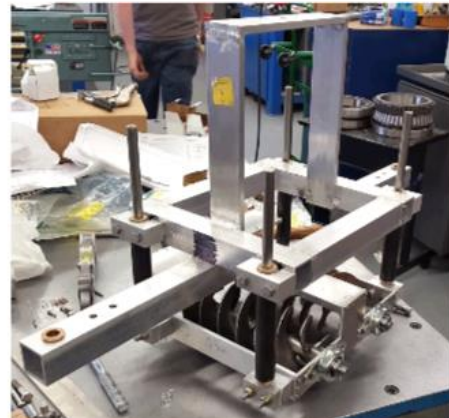
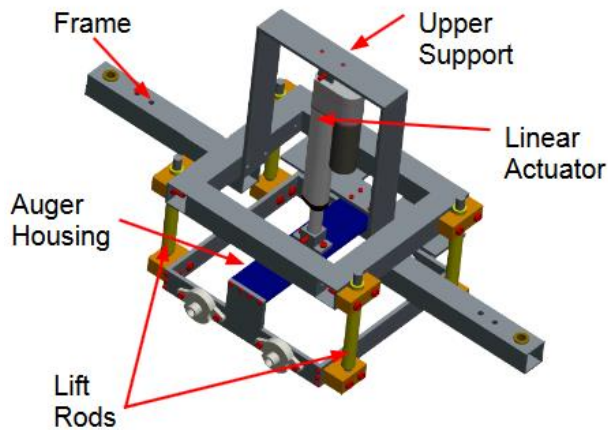


Figure 4: CAD and actual Robo-Weeder with frame and lift subassembly connected to auger housing.

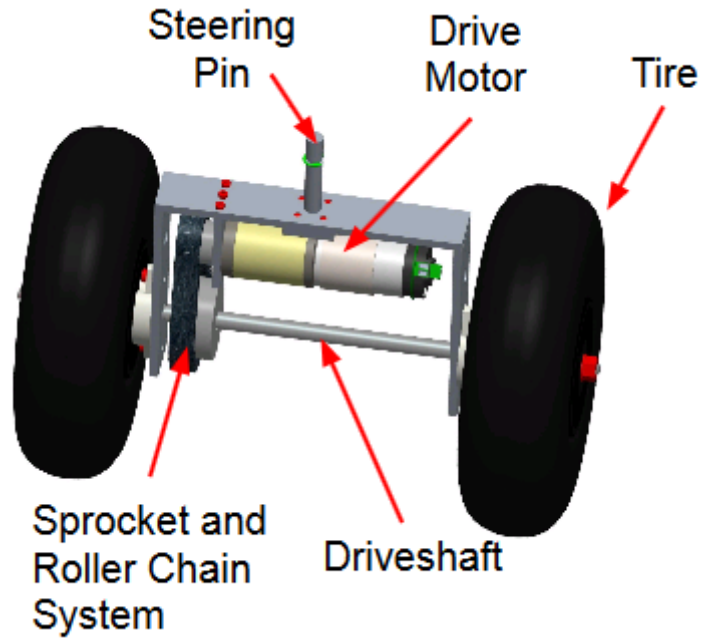


Figure 5: Drive subassembly with labels.

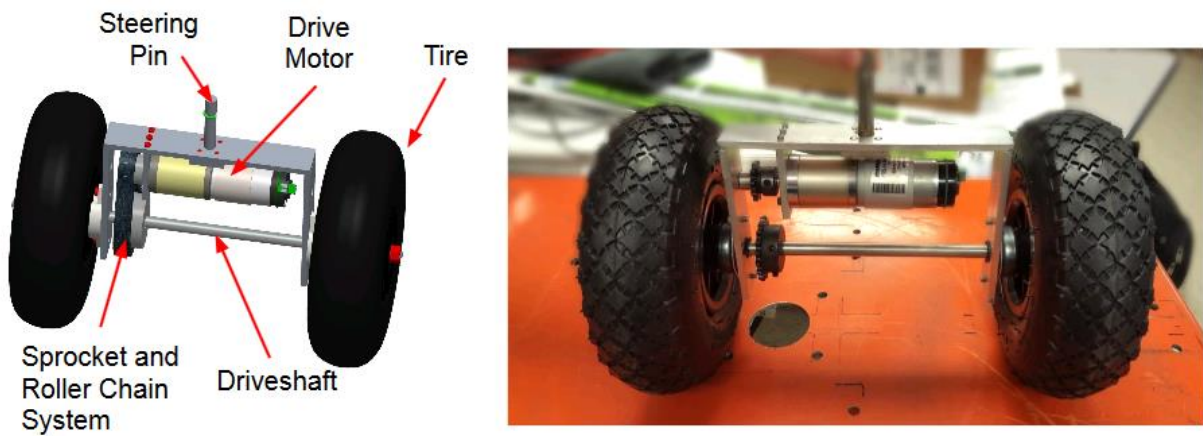


Figure 6: CAD and Robo-Weeder's drive assembly.

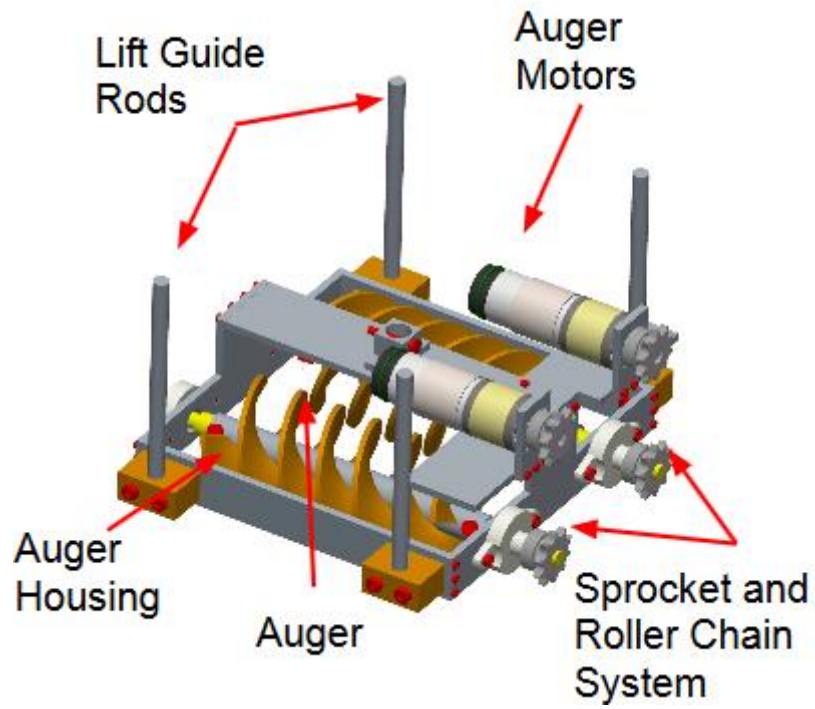


Figure 7: Auger subassembly with labels.

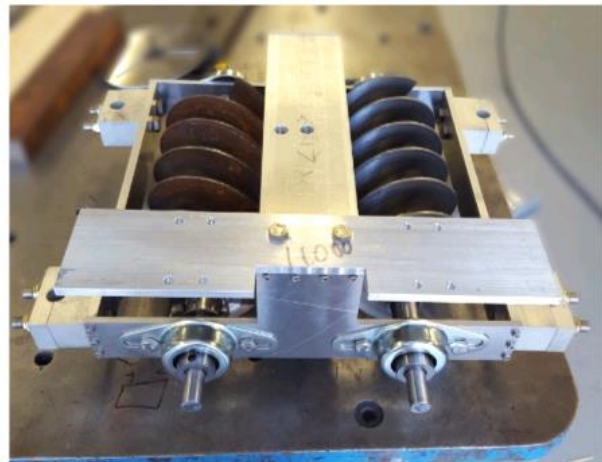
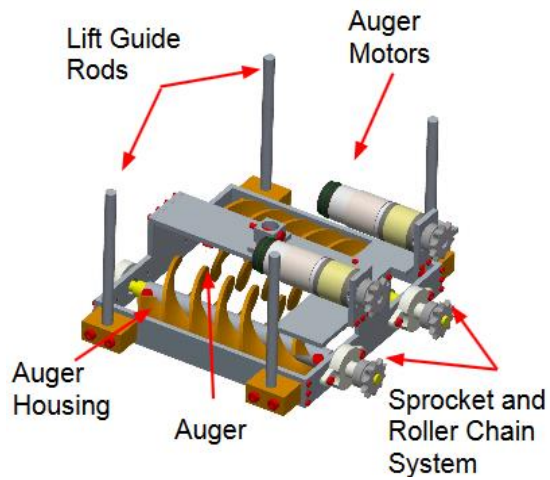


Figure 8: CAD and Robo-Weeder's auger subassembly.

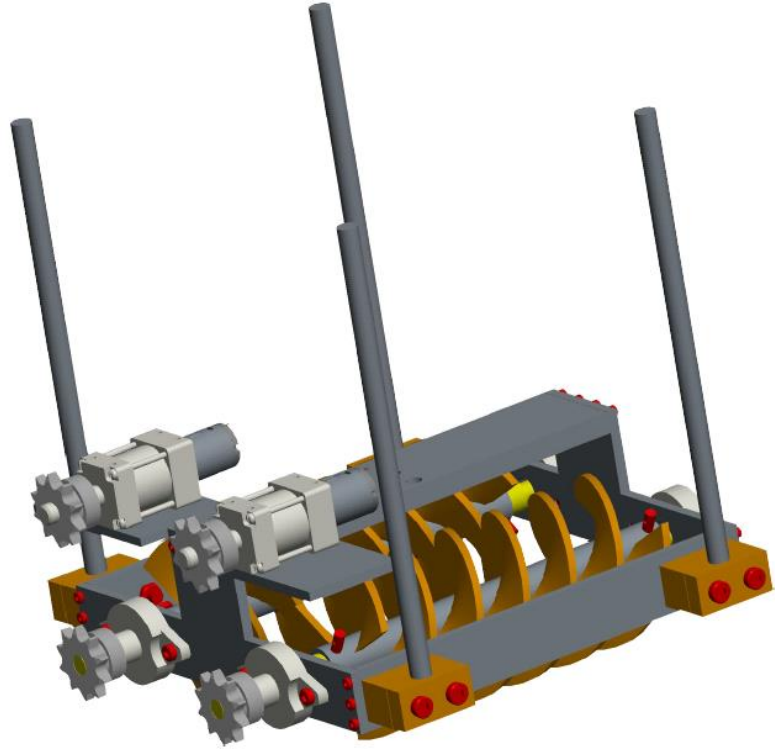


Figure 9: Auger subassembly.

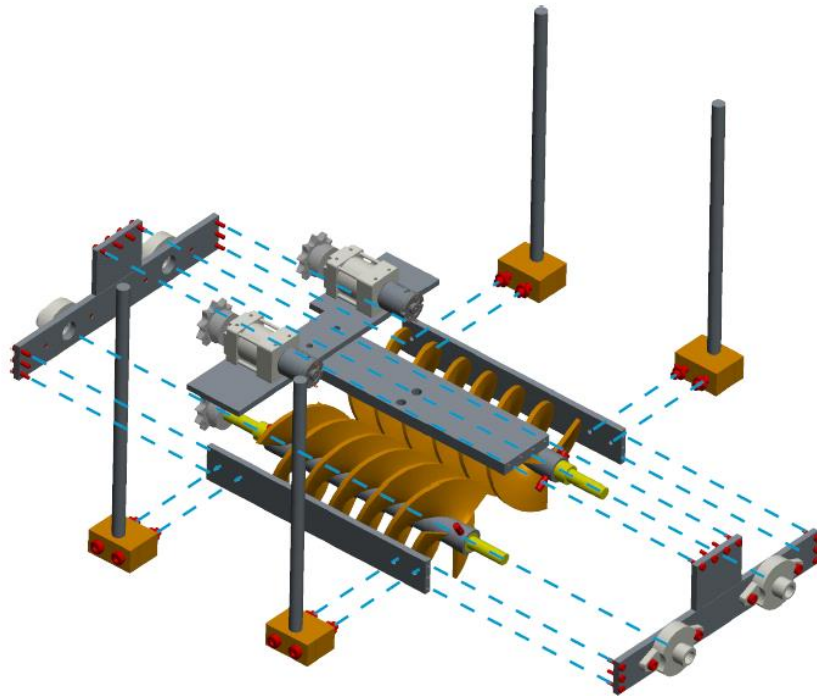


Figure 10: Auger subassembly dismantled.

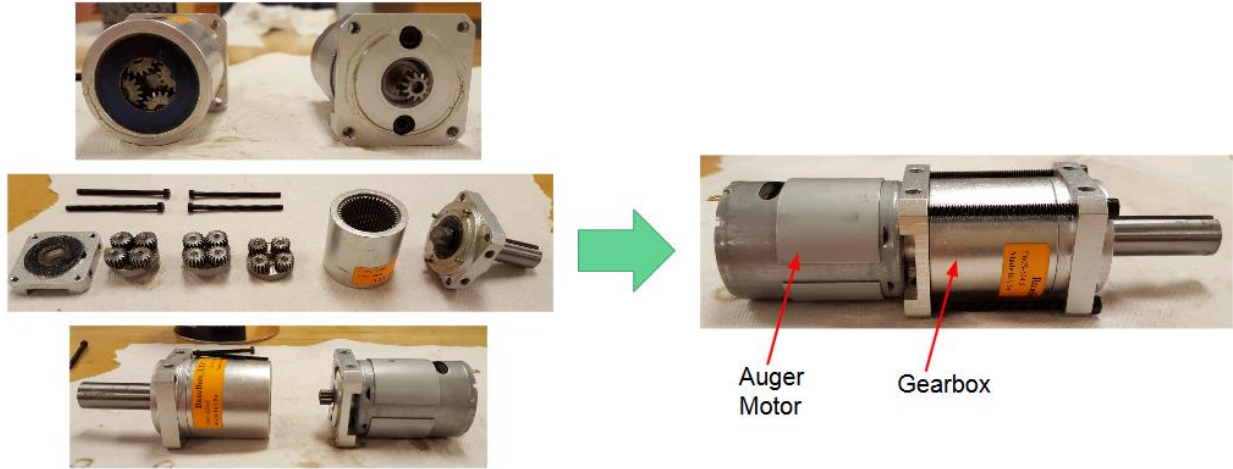


Figure 11: Auger motor and planetary gearbox coupling.

Appendix B

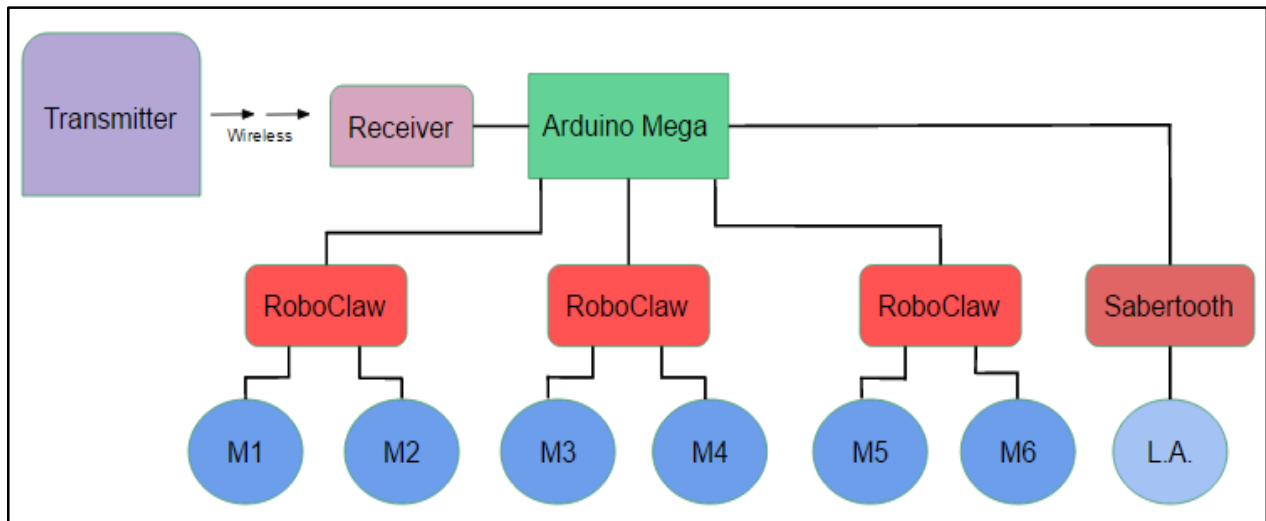


Figure 1: Design flow from user to motors

Appendix C

```
/* Robo-Weeder
 * Master Code
 */
int analogOutput = 0;          //#####FRONT- Steering
int encoder_value = 0;
int n = 0; //counter
int rc = 0;
int A = 0; // Encoder spin
int a = 0;
int b = 0;
int c = 0;
int d = 0;
int e = 0;
int pulse = 0;
int z = 0; // reference turn
int ref = 0;
int Rcounter = 0;
int Lcounter = 0;
int x = 0; // counter stop
int CC = 0;
int CCW = 0;
int count = 0;
int ReturnCounter = 0;
int CR = 0; //turn completion left side
int CL = 0; //turn completion right side
//int pulse = 0; //counter pulse

int B = 0;
int C = 0;
int cc = 40; //center

int rcpin1 = 31; // input pin from rc receiver channel 1 Front Steering
int rcpin2 = 33; // input pin from rc receiver channel 2 Drive
int rcpin4 = 35; // input pin from rc receiver channel 4 Rear Steering
int rcpin5 = 37; // input pin from rc receiver channel 5 Auger
int rcpin6 = 39; // input pin from rc receiver channel 6 Linear Actuator

#include <Servo.h>
#define MIN 1410
#define MAX 1590
#define STOP 1500
Servo myservo1; // create servo object to control a RoboClaw channel
int ch1; //Variables to store and display the values of each channel
int mapCh1; //ch1 mapped to 255
```



```

int analogOutputR = 0;          //#####REAR- Steering
int encoder_valueR = 0;
int nR = 0; //counter
int rcR = 0;
int AR = 0; // Encoder spin
int aR = 0;
int bR = 0;
int cR = 0;
int dR = 0;
int eR = 0;
int pulseR = 0;
int zR = 0; // reference turn
int refR = 0;
int RcounterR = 0;
int LcounterR = 0;
int xR = 0; // counter stop
int CCR = 0;
int CCWR = 0;
int countR = 0;
int ReturnCounterR = 0;
int CRR = 0; //turn completion left side
int CLR = 0; //turn completion right side

int BR = 0;
//int C = 0;
int ccR = 40; //center

Servo myservo2; // create servo object to control a RoboClaw channel
int ch4; //Variables to store and display the values of each channel
int mapCh4; //ch4 mapped to 255

//#####DRIVE
#include <Servo.h>
#define MIN 1400
#define STOP 1500
#define MAX1 1600
#define MAX2 1675
#define MAX3 1750
Servo myservo3; // Drive Motor 1
Servo myservo4; // Drive Motor 2
int rcpinD = 33; // input pin from rc receiver
int posd = 0; // variable to store the servo position
int ch2; //Variables to store and display the values of each channel

#define encoderOPinAD 2 // figure out mega interrupt
#define encoderOPinBD 3 // figure out mega interrupt

```

```

volatile long encoder0PosD=0;
long newpositionD;
long oldpositionD = 0;
unsigned long newtimeD;
unsigned long oldtimeD = 0;
long veld;

//#####AUGER
Servo myservo5; // Drive Motor 1
Servo myservo6; // Drive Motor 2
int rcpin = 37; // input pin from rc receiver
int pos = 0; // variable to store the servo position
int ch5; //Variables to store and display the values of each channel

#define encoder0PinA 18 // figure out mega interrupt
#define encoder0PinB 19 // figure out mega interrupt

volatile long encoder0Pos=0;
long newposition;
long oldposition = 0;
unsigned long newtime;
unsigned long oldtime = 0;
long vel;

void setup() //#####VOID SETUP
{
pinMode(A1, INPUT); //Encoder front Steering #####STEERING
pinMode(rcpin1, INPUT);
myservo1.attach(4); // attaches the RC signal on pin 2 to the servo object
pinMode(A2, INPUT); // pinMode(encoder_pin, INPUT); //Encoder rear Steering
pinMode(rcpin4, INPUT);
myservo2.attach(5); // attaches the RC signal on pin 3 to the servo object
Serial.begin(9600);

pinMode(encoder0PinAD, INPUT); //#####DRIVE
digitalWrite(encoder0PinAD, HIGH); // turn on pullup resistor
pinMode(encoder0PinBD, INPUT);
digitalWrite(encoder0PinBD, HIGH); // turn on pullup resistor
attachInterrupt(0, doEncoderD, RISING); // encoDER ON PIN 2 3

pinMode(rcpinD, INPUT);
myservo2.attach(6); // attaches the RC signal on pin 4 to the servo object
myservo3.attach(7); // attaches the RC signal on pin 5 to the servo object

pinMode(encoder0PinA, INPUT); //#####AUGER
digitalWrite(encoder0PinA, HIGH); // turn on pullup resistor
pinMode(encoder0PinB, INPUT);
digitalWrite(encoder0PinB, HIGH); // turn on pullup resistor

```



```

        a = 0;
        b = 0;
        n = 0;
        x = 0; }
    }
    Serial.print ("CC: ");
    Serial.println (count) ;
} }
if (CCW == 1)    //#####Counter-ClockWise
{
if ((A >= 0) && (A <= 1024))    // CENTER
{
if ((A > 0) && (A <= 512))
{ a = a + 1;

if ((a == 2) && (x != 1))
{ count = count - 1;
a = 0;
x = 1; // counter stop when rotation stop
} else if ((a >= 1) && (b >= 1))
{ count = count - 1;
a = 0;
b = 0;
n = 0;
x = 0; }

}
if ((A > 512) && (A <= 1024))
{ b = b + 1;
if ((x != 1) && (b == 2))
{ count = count - 1;
b = 0;
x = 1; // counter stop
} else if ((a >= 1) && (b >= 1))
{ count = count - 1;
a = 0;
b = 0;
n = 0;
x = 0; }

}
Serial.print ("CCW: ");
Serial.println (count) ;
} }
if (ch1 == 0)
{ myservo1.writeMicroseconds(STOP); //stop
}
if ((ch1 >1400) && (ch1 <1505))    //#####CENTER
{ CR = 1;

```

```

CL = 1;
if (cc != 40)
{ z = 40 - ref; //calculate logic
  if (z < 0)
    { CCW = 1; //CCW countdown
      myservo1.writeMicroseconds(MIN);
      Serial.print ("ReturnCounter: ");
      Serial.println (count) ;

      if (count == 0) // pulse = 20, 3 pulse/rev (20/3 = 10 degrees)
        { myservo1.writeMicroseconds(STOP);
          ReturnCounter = 0;
          CCW = 0; //CCW countdown
          cc = 40; //transmitter position
          ref = 40; //pulse reference
        } }
    else
      { CCW = 1; //CCW countdown
        myservo1.writeMicroseconds(MAX);
        Serial.print ("ReturnCounter: ");
        Serial.println (count) ;

        if (count == 0) // pulse = 20, 3 pulse/rev (20/3 = 10 degrees)
          { myservo1.writeMicroseconds(STOP);
            CCW = 0; //CCW countdown
            cc = 40; //transmitter position
            ref = 40; //pulse reference
          }
        }

} else { myservo1.writeMicroseconds(STOP); // stop
        count = 0; } //##### manual steering

}
//
if ((ch1 >= 1505) && (ch1 <2000) && (CR ==1)) //#####RIGHT TURN
{ if (cc != 50)
  { z = 50 - ref;
    CC = 1;
    myservo1.writeMicroseconds(MAX);
    Serial.print ("Right Turn: ");
    Serial.println (count) ;

    if (count == 5) // pulse = 20, 3 pulse/rev (20/3 = 10 degrees)
      { myservo1.writeMicroseconds(STOP);
        // ReturnCounter = 18;
        // count = 0;
        CC = 0;
      }
    }
}

```

```

        cc = 50;
        CL = 1; //Turn left enable
        ref = 50; //pulse reference
    } }
    else {myservo1.writeMicroseconds(STOP); }
}
if ((ch1 >= 950) && (ch1 <= 1400) && (CL == 1)) //#####LEFT TURN
{ if (cc != 30)
    { z = 30 - ref;
      CC = 1;
      myservo1.writeMicroseconds(MIN);
      Serial.print ("Left Turn: ");
      Serial.println (count) ;

      if (count == 5) // pulse = 20, 3 pulse/rev (20/3 = 10 degrees)
      { myservo1.writeMicroseconds(STOP);
        // ReturnCounter = 18;
        // count = 0;
        CC = 0;
        cc = 30;
        CR = 1; //Turn right enable
        ref = 30; //pulse reference
      } }
    else { myservo1.writeMicroseconds(STOP); }
}
analogOutputR = (analogRead(A2)); //encoder ADC analog 1024##### REAR LOOP
AR = analogOutputR;
encoder_valueR = map(analogOutputR, 0, 1024, 0, 255); // encoder_value = pulseIn(5, HIGH);
Serial.print("EncoderR: ");
Serial.print(AR); //encoder ADC analog
Serial.write(10); //ascii line feed
ch4 = pulseIn (rcpin4,HIGH); //Read and store channel 1

Serial.print ("Ch4:"); //Display text string on Serial Monitor to distinguish variables
Serial.println (ch4); //Print in the value of channel 1
if (CCR == 1) //#####ClockWise
{
if ((AR >= 0) && (AR <= 1024)) // CENTER
{
if ((AR > 0) && (AR <= 512))
{ aR = aR + 1;

if ((aR == 2) && (xR != 1)) // will not stop counting when not running
{ countR = countR + 1;
aR = 0;
xR = 1; // counter stop when rotation stop
} else if ((aR >= 1) && (bR >= 1))
{ countR = countR + 1;

```

```

        aR = 0;
        bR = 0;
        nR = 0;
        xR = 0; }

}
if ((AR > 512) && (AR <= 1024))
{ bR = bR + 1;
  if ((xR != 1) && (bR == 2))
  { countR = countR + 1;
    bR = 0;
    xR = 1; // counter stop
  } else if ((aR >= 1) && (bR >= 1))
  { countR = countR + 1;
    aR = 0;
    bR = 0;
    nR = 0;
    xR = 0; }
}

    Serial.print ("CCR: ");
    Serial.println (countR) ;
} }
if (CCWR == 1)    //#####Counter-ClockWise
{
if ((AR >= 0) && (AR <= 1024))    // CENTER
{
if ((AR > 0) && (AR <= 512))
{ aR = aR + 1;

    if ((aR == 2) && (xR != 1))
    { countR = countR - 1;
      aR = 0;
      xR = 1; // counter stop when rotation stop
    } else if ((aR >= 1) && (bR >= 1))
    { countR = countR - 1;
      aR = 0;
      bR = 0;
      nR = 0;
      xR = 0; }

}
if ((AR > 512) && (AR <= 1024))
{ bR = bR + 1;
  if ((xR != 1) && (bR == 2))
  { countR = countR - 1;
    bR = 0;
    xR = 1; // counter stop
  } else if ((aR >= 1) && (bR >= 1))

```

```

    { countR = countR - 1;
      aR = 0;
      bR = 0;
      nR = 0;
      xR = 0; }
  }
  Serial.print ("CCWR: ");
  Serial.println (countR) ;
} }
if (ch4 == 0)
  { myservo2.writeMicroseconds(STOP); //stop
  }
if ((ch4 >1400) && (ch4 <1505))    //#####CENTER
  { CRR = 1;
    CLR = 1;
    if (ccR != 40)
      { zR = 40 - refR; //calculate logic
        if (zR < 0)
          { CCWR = 1; //CCW countdown
            myservo2.writeMicroseconds(MIN);
            Serial.print ("ReturnCounterR: ");
            Serial.println (countR) ;

            if (analogOutputR == analogRead(A2)) //encoder ADC analog 1024
          { AR = analogOutputR;
            encoder_valueR = map(analogOutputR, 0, 1024, 0, 255); // encoder_value = pulseIn(5, HIGH);
            Serial.print("EncoderR: ");
            Serial.print(AR); //encoder ADC analog
            Serial.write(10); //ascii line feed
            ch1 = pulseIn (rcpin4,HIGH); //Read and store channel 1

            Serial.print ("Ch4:"); //Display text string on Serial Monitor to distinguish variables
            Serial.println (ch4); //Print in the value of channel 1
          if (CCR == 1)    //#####ClockWise
          {
            if ((AR >= 0) && (AR <= 1024))    // CENTER
            {
              if ((AR > 0) && (AR <= 512))
                { aR = aR + 1;

                  if ((aR == 2) && (xR != 1))    // will not stop counting when not running
                    { countR = countR + 1;
                      aR = 0;
                      xR = 1; // counter stop when rotation stop
                    } else if ((aR >= 1) && (bR >= 1))
                      { countR = countR + 1;
                        aR = 0;
                        bR = 0;

```



```

        nR = 0;
        xR = 0; }

}
if ((AR > 512) && (AR <= 1024))
{ bR = bR + 1;
  if ((xR != 1) && (bR == 2))
  { countR = countR + 1;
    bR = 0;
    xR = 1; // counter stop
  } else if ((aR >= 1) && (bR >= 1))
  { countR = countR + 1;
    aR = 0;
    bR = 0;
    nR = 0;
    xR = 0; }
}

    Serial.print ("CCR: ");
    Serial.println (countR);
} }
if (CCWR == 1) //#####Counter-ClockWise
{
if ((AR >= 0) && (AR <= 1024)) // CENTER
{
if ((AR > 0) && (AR <= 512))
{ aR = aR + 1;

    if ((aR == 2) && (xR != 1))
    { countR = countR - 1;
      aR = 0;
      xR = 1; // counter stop when rotation stop
    } else if ((aR >= 1) && (bR >= 1))
    { countR = countR - 1;
      aR = 0;
      bR = 0;
      nR = 0;
      xR = 0; }

}
if ((AR > 512) && (AR <= 1024))
{ bR = bR + 1;
  if ((xR != 1) && (bR == 2))
  { countR = countR - 1;
    bR = 0;
    xR = 1; // counter stop
  } else if ((aR >= 1) && (bR >= 1))
  { countR = countR - 1;
    aR = 0;

```

```

        bR = 0;
        nR = 0;
        xR = 0; }
    }
        Serial.print ("CCWR: ");
        Serial.println (countR) ;
} }
if (ch4 == 0)
{ myservo2.writeMicroseconds(STOP); //stop
}
if ((ch1 >1400) && (ch1 <1505)) //#####CENTER
{ CR = 1;
  CL = 1;
  if (cc != 40)
  { z = 40 - ref; //calculate logic
    if (z < 0)
    { CCW = 1; //CCW countdown
      myservo1.writeMicroseconds(MIN);
      Serial.print ("ReturnCounter: ");
      Serial.println (count) ;

      if (count == 0) // pulse = 20, 3 pulse/rev (20/3 = 10 degrees)
      { myservo1.writeMicroseconds(STOP);
        ReturnCounter = 0;
        CCW = 0; //CCW countdown
        cc = 40; //transmitter position
        ref = 40; //pulse reference
      } }
    else
    { CCW = 1; //CCW countdown
      myservo1.writeMicroseconds(MAX);
      Serial.print ("ReturnCounter: ");
      Serial.println (count) ;

      if (count == 0) // pulse = 20, 3 pulse/rev (20/3 = 10 degrees)
      { myservo1.writeMicroseconds(STOP);
        CCW = 0; //CCW countdown
        cc = 40; //transmitter position
        ref = 40; //pulse reference
      }
    }

  } else { myservo1.writeMicroseconds(STOP); // stop
    count = 0; } //##### manual steering
}

if ((ch1 >= 1505) && (ch1 <2000) && (CR == 1)) //#####RIGHT TURN

```

```

{ if (cc != 50)

    { z = 50 - ref;
      CC = 1;
      myservo1.writeMicroseconds(MAX);
      Serial.print ("Right Turn: ");
      Serial.println (count) ;

      if (count == 5) // pulse = 20, 3 pulse/rev (20/3 = 10 degrees)
        { myservo1.writeMicroseconds(STOP);
          // ReturnCounter = 18;
          // count = 0;
          CC = 0;
          cc = 50;
          CL = 1; //Turn left enable
          ref = 50; //pulse reference
        } }
      else {myservo1.writeMicroseconds(STOP); }
    }
if ((ch1 >= 950) && (ch1 <= 1400) && (CL == 1)) //#####LEFT TURN
{ if (cc != 30)
  { z = 30 - ref;
    CC = 1;
    myservo1.writeMicroseconds(MIN);
    Serial.print ("Left Turn: ");
    Serial.println (count) ;

    if (count == 5) // pulse = 20, 3 pulse/rev (20/3 = 10 degrees)
      { myservo1.writeMicroseconds(STOP);
        // ReturnCounter = 18;
        // count = 0;
        CC = 0;
        cc = 30;
        CR = 1; //Turn right enable
        ref = 30; //pulse reference
      } }
    else { myservo2.writeMicroseconds(STOP); }

    { myservo2.writeMicroseconds(STOP);
      ReturnCounterR = 0;
      CCWR = 0; //CCW countdown
      ccR = 40; //transmitter position
      refR = 40; //pulse reference
    } }
  else
  { CCWR = 2; //CCW countdown
    myservo2.writeMicroseconds(MAX);
    Serial.print ("ReturnCounterR: ");

```

```

Serial.println (countR) ;

if (countR == 0) // pulse = 20, 3 pulse/rev (20/3 = 10 degrees)
{ myservo2.writeMicroseconds(STOP);
  CCWR = 0; //CCW countdown
  ccR = 40; //transmitter position
  refR = 40; //pulse reference
}
}

} else { myservo2.writeMicroseconds(STOP); // stop
        countR = 0; } //##### manual steering

}

if ((ch4 >= 1505) && (ch4 <2000) && (CR == 1)) //#####RIGHT TURN
{ if (ccR != 50)
  { zR = 50 - refR;
    CCR = 1;
    myservo2.writeMicroseconds(MAX);
    Serial.print ("Right TurnR: ");
    Serial.println (countR) ;

    if (countR == 5) // pulse = 20, 3 pulse/rev (20/3 = 10 degrees)
    { myservo2.writeMicroseconds(STOP);
      CCR = 0;
      ccR = 50;
      CLR = 1; //Turn left enable
      refR = 50; //pulse reference
    } }
  else {myservo2.writeMicroseconds(STOP); }
}

if ((ch4 >= 950) && (ch4 <= 1400) && (CL == 1)) //#####LEFT TURN
{ if (ccR != 30)
  { zR = 30 - refR;
    CCR = 1;
    myservo2.writeMicroseconds(MIN);
    Serial.print ("Left TurnR: ");
    Serial.println (countR) ;

    if (countR == 5) // pulse = 20, 3 pulse/rev (20/3 = 10 degrees)
    { myservo2.writeMicroseconds(STOP);
      CCR = 0;
      ccR = 30;
      CRR = 1; //Turn right enable
      refR = 30; //pulse reference
    } }
  else { myservo2.writeMicroseconds(STOP); }
}
}

```

```

}}
newpositionD = encoder0PosD; //#####LOOP DRIVE
newtimeD = millis();
velD = (newpositionD-oldpositionD) * 1000 /(newtimeD-oldtimeD);
Serial.print ("speed = ");
Serial.println (velD);
oldpositionD = newpositionD;
oldtimeD = newtimeD;
delay(250);
ch2 = pulseIn (rcpin2,HIGH); //Read and store channel 2
Serial.print ("Ch2:");
Serial.println (ch2);

if (ch2 == 0)
  { myservo3.writeMicroseconds(STOP);
    myservo4.writeMicroseconds(STOP); } //stop
if ((ch2 >1475) && (ch2 <1505))
  { myservo3.writeMicroseconds(STOP);
    myservo4.writeMicroseconds(STOP);} //stop
if ((ch2 >= 1505) && (ch2 <1700))
  { myservo3.writeMicroseconds(MAX1);
    myservo4.writeMicroseconds(MAX1); } // forward 30%
if ((ch2 >= 1700) && (ch2 <1850) )
  { myservo3.writeMicroseconds(MAX2);
    myservo4.writeMicroseconds(MAX2); } // forward 60%
if ((ch2 >= 1850) && (ch2 <2000) )
  { myservo2.writeMicroseconds(MAX3);
    myservo3.writeMicroseconds(MAX3); } // forward 100%
if ((ch2 > 0) && (ch2 <= 1475))
  { myservo3.writeMicroseconds(MIN);
    myservo4.writeMicroseconds(MIN); } //full reverse

newposition = encoder0Pos; //#####LOOP AUGER
newtime = millis();
vel = (newposition-oldposition) * 1000 /(newtime-oldtime);
Serial.print ("speed = ");
Serial.println (vel);
oldposition = newposition;
oldtime = newtime;
delay(250);
ch5 = pulseIn (rcpin5,HIGH); //Read and store channel 5
Serial.print ("Ch5:");
Serial.println (ch5);

if (ch5 == 0)
  { myservo5.writeMicroseconds(STOP);
    myservo6.writeMicroseconds(STOP);} //stop
if ((ch5 >1475) && (ch5 <1505))

```

```

    { myservo5.writeMicroseconds(STOP);
      myservo6.writeMicroseconds(STOP);} //stop
if ((ch5 >= 1505) && (ch5 <1700))
  { myservo5.writeMicroseconds(MAX1);
    myservo6.writeMicroseconds(MAX1);} // forward 30%
if ((ch5 >= 1700) && (ch5 <1850) )
  { myservo5.writeMicroseconds(MAX2);
    myservo6.writeMicroseconds(MAX2);} // forward 60%
if ((ch5 >= 1850) && (ch5 <2000) )
  { myservo5.writeMicroseconds(MAX3);
    myservo6.writeMicroseconds(MAX3);} // forward 100%
if ((ch5 > 0) && (ch5 <= 1475))
  { myservo5.writeMicroseconds(MIN);
    myservo6.writeMicroseconds(MIN);} //full reverse
} //#####LOOP BODY

void doEncoderD() //#####InterruptCall DRIVE
{
if (digitalRead(encoder0PinAD) == digitalRead(encoder0PinBD)) {
  encoder0PosD++;
} else {
  encoder0PosD--;
} }
void doEncoder() //#####InterruptCall AUGER
{
if (digitalRead(encoder0PinA) == digitalRead(encoder0PinB)) {
  encoder0Pos++;
} else {
  encoder0Pos--;
} }

```