MCT331

DESIGN OF MECHATRONICS SYSTEMS

TEAM 2

Prepared By:

Ahmed Hassabou

Mohamed Fathi

Abdel Rahman Habash

AIN SHAMS UNIVERSITY
FACULTY OF ENGINEERING
International Credit Hours Engineering Programs (i.CHEP)

MCT331

Design of Mechatronics (1)

Final Submission

Supervised by

Dr. Shady Ahmed Maged

Eng. Michael Sami Hannalla

Eng. Ahmed Khaled Mohamed

Eng. Mohamed Nabil Mohamed

Eng. Ahmed Hussein Mohamed

Eng. Hossam Moataz El-Keshky

Eng. Mohamed Usama Mohamed

Submitted by

Ahmed Hisham Fathy Hassabou 19P4007

Mohamed Fathi Abdelhamid Ali 19P4401

Abdel Rahman Ahmed Habash 19P5553

Table of Contents

TABLE OF FIGURES

ABSTRACT

Feeding, Sorting, and Storing. These words simplifies what was required from us, a production line that feed products from a magazine towards the sorting terminal and then according to the differences between products each category will have a specific storing locations, at this point comes the role of the 3 degree of freedom cartesian robot integrated with suction components to grab the product, move towards its storage location and releases it. Firstly, the overall line components were drawn using SOLIDWORKS with the planned materials and dimensions. Secondly, simulation of the integrated processes of the line was generated using MATLAB Simscape and Simulink Libraries to ensure that the planned mechanical components and idea of actuation will meet the requirements. Furthermore, stress analysis and actuator sizing calculations were done and components were chosen according to the calculations done. After the mechanical part was done, it comes to the programming one. As required, our project is based on RTOS (Real Time Operating System), and microcontrollers communicate together through ethernet connection. Finally, we can say that the harder you work the better you get. For actuation video: MCT331_Final_Project - Google Drive

1.0 INTRODUCTION

Simulation allows you to investigate hypothetical situations rather than performing system-based experiments. Its assists you in identifying obstructions in the flow of materials, information, and process. In addition to that, it allows you to gain insight into which variables are most important to system performance. Getting the big picture and fixing minor and major problems before the implementation process is a more superior and efficient way in integration.

So, after knowing the problems we were going to face, we were able to produce the line and assemble it taking all mechanical considerations insight. And beyond the mechanical part, we were able to actuate the line according to what was required, more details will be described in this report.

2.0 MECHANICAL DESIGN

2.1 CAD Design:

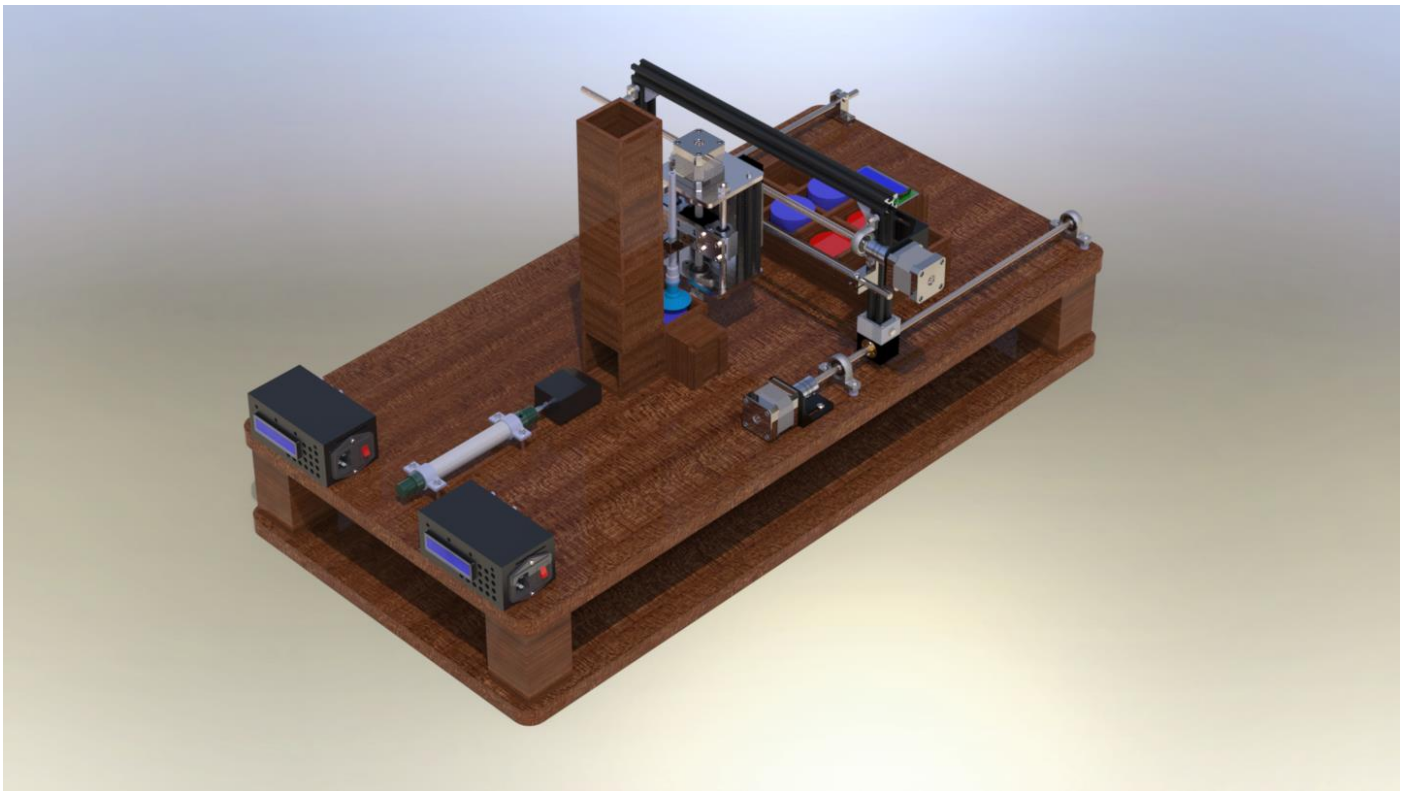


Figure 1: Production Line Isometric View

Our production line is designed to fit for 8 products that are divided into 2 categories, large and small products. Feeding mechanism is based on the famous 2-way pneumatic piston cylinder. The cartesian robotic arm contains 3 axis that are based on the lead screw with guide nut connection. The production line contains 2 control boxes, one for the electrical components and the other for the pneumatic ones.

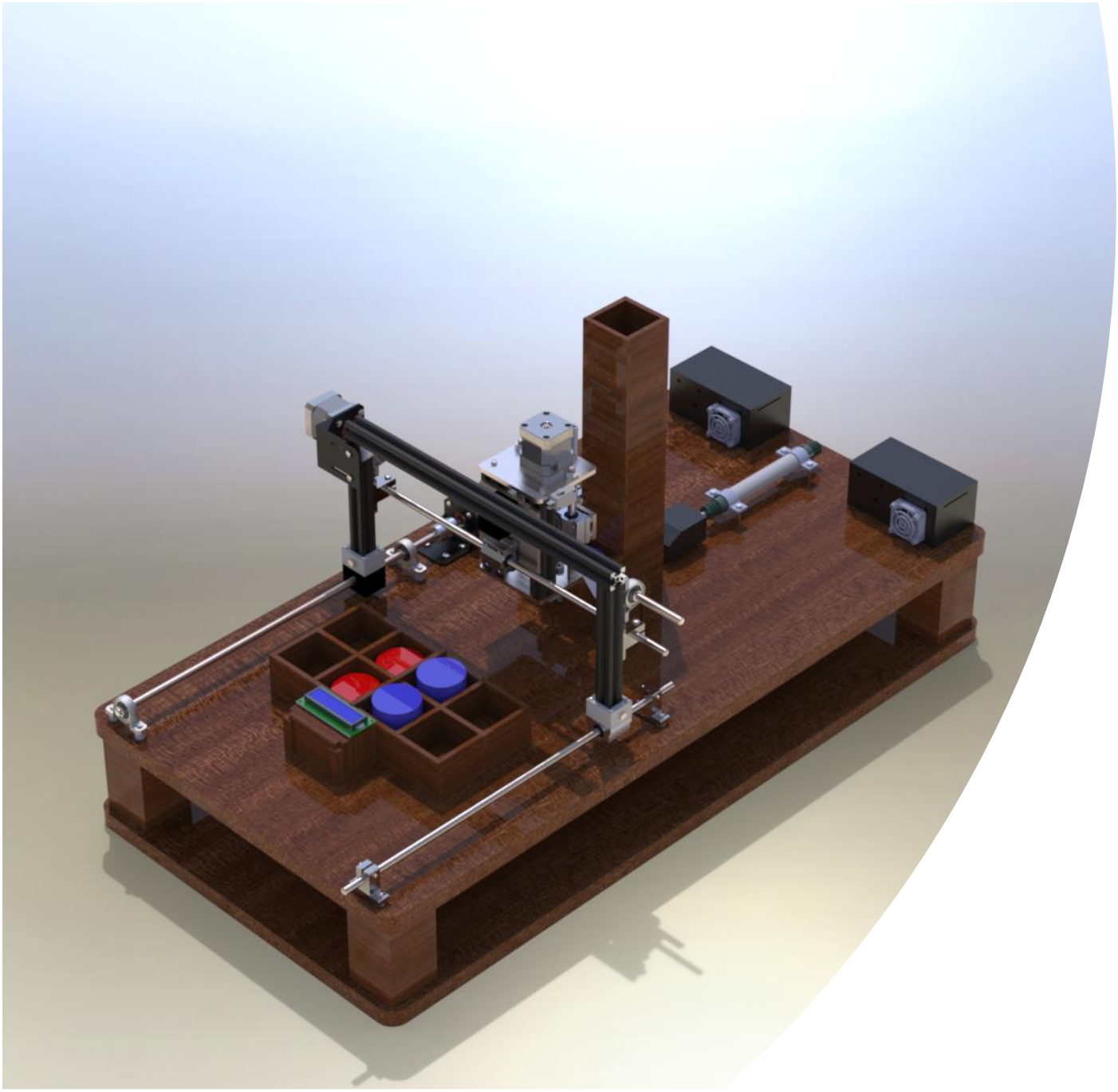


Figure 2: Production Line Reverse Isometric View

Storage terminal provide locations for 8 products, 4 of each category, and it comes with a LCD that displays the count of each category available in the storage.

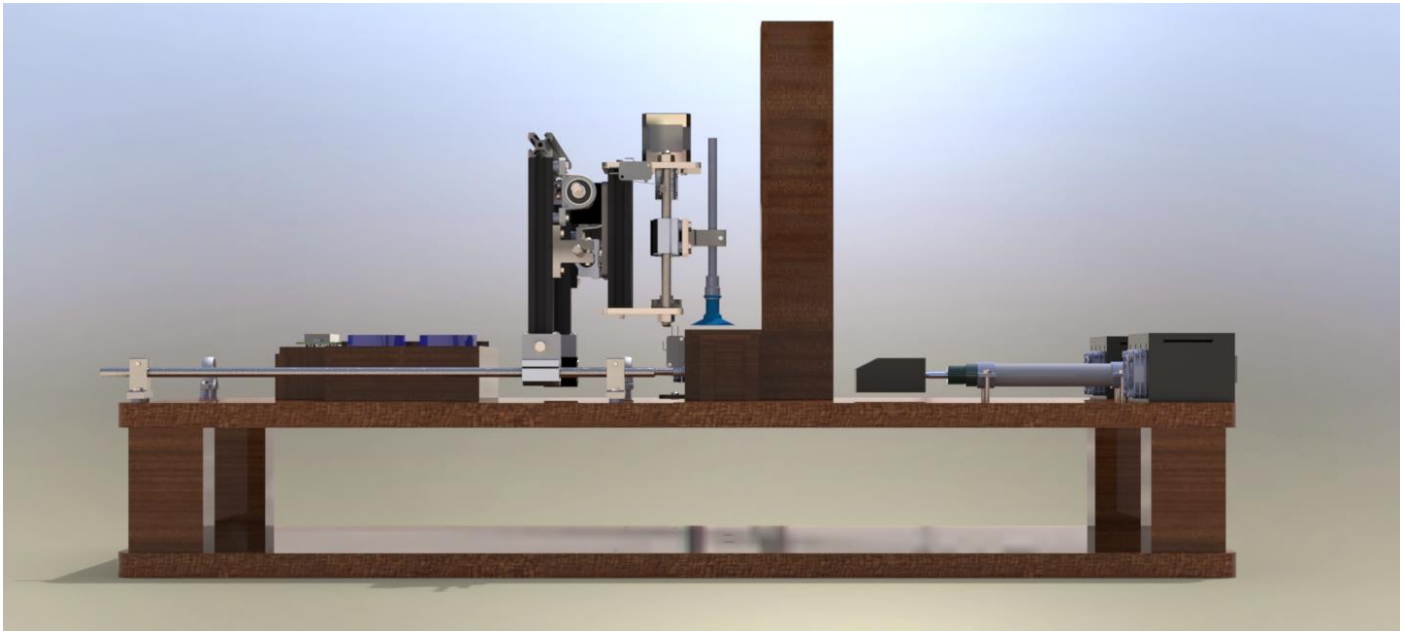


Figure 3: Production Line Side Open View

The Production Line base is made from 2 levels so that all wirings are not seen and are easily accessible by removing the Side Cover as in figure 3. The default view of the line is shown in figure 4.

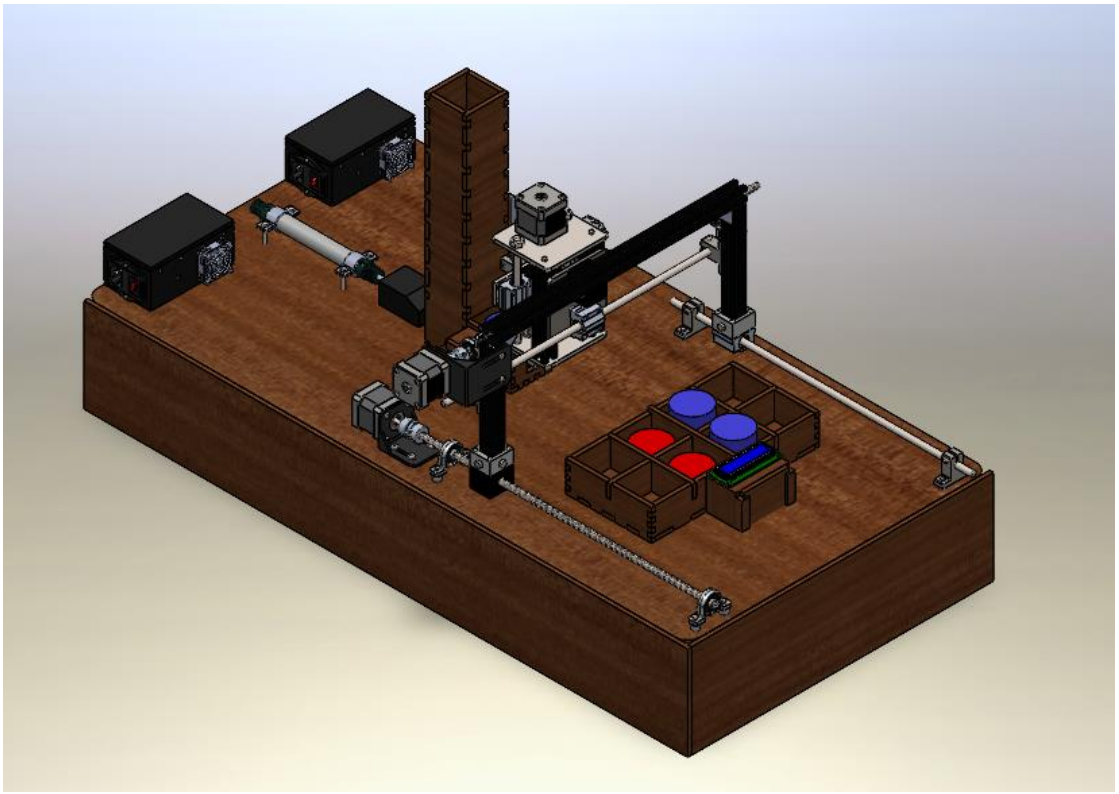


Figure 4: Production Line Side Closed View

2.2 Actuator Sizing

2.2.1. Mass calculations

2.2.1.1. Material density

$\rho_{Al} = 2710\ kg/m^3$

$\rho_{st} = 7500\ kg/m^3$

$\rho_{Al} = 1210\ kg/m^3$

2.2.1.2. Leg

$m = \rho V = 2710 \times (160 \times 20 \times 20) \times 10^{-9} = 0.17344\ Kg$

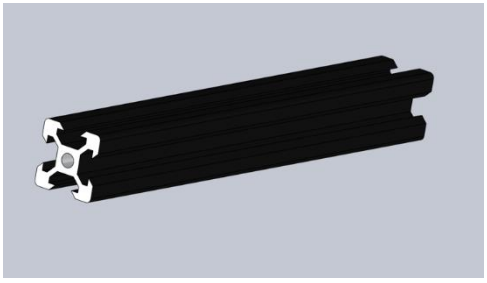


Figure 5: Robotic Arm Leg

2.2.1.3. Y-axis Aluminum rod

$m = \rho V = 2710 \times (380 \times 20 \times 20) \times 10^{-9} = 0.41192\ Kg$

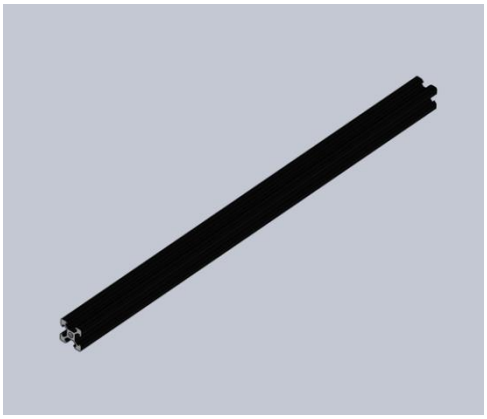


Figure 6: Y-Axis Aluminium Rod

2.2.1.4. Y-axis linear bearing rod

$$m = \rho V = 2710 \times \pi \times (8)^2 \times 450 \times 10^{-9} = 0.245\ Kg$$

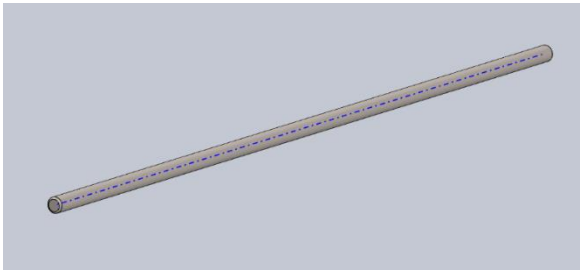


Figure 7: Y-Axis Linear Rod

2.2.1.5. Y-axis power screw

$$m = \rho V = 7500 \times \pi \times (8)^2 \times 500 \times 10^{-9} = 0.754\ Kg$$

Figure 8: Y-Axis Power Screw

2.2.1.6. Z-axis aluminum rod

$$m = \rho V = 2710 \times (120 \times 20 \times 20) \times 10^{-9} = 0.13\ Kg$$

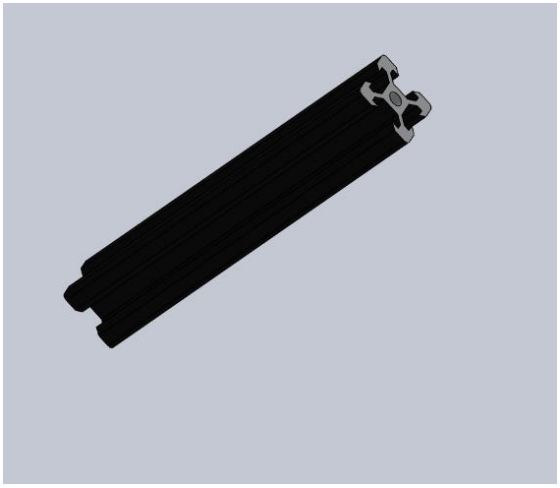


Figure 9: Z-Axis Aluminum Rod

2.2.1.7. Z-axis plate

$$m = \rho V = 2710 \times (84 \times 84 \times 5) \times 10^{-9} = 0.09561\ Kg$$

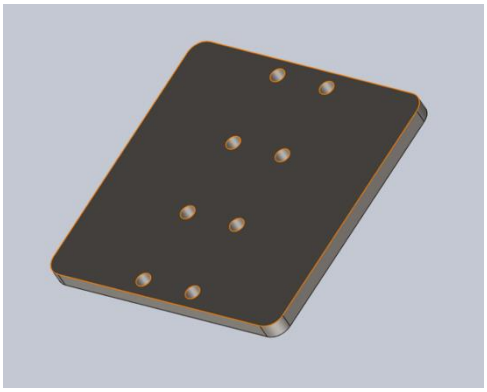


Figure 10: Z-Axis Plate

2.2.1.8. Z-axis linear bearing rod

$$m = \rho V = 2710 \times \pi \times (8)^2 \times 150 \times 10^{-9} = 0.082\ Kg$$

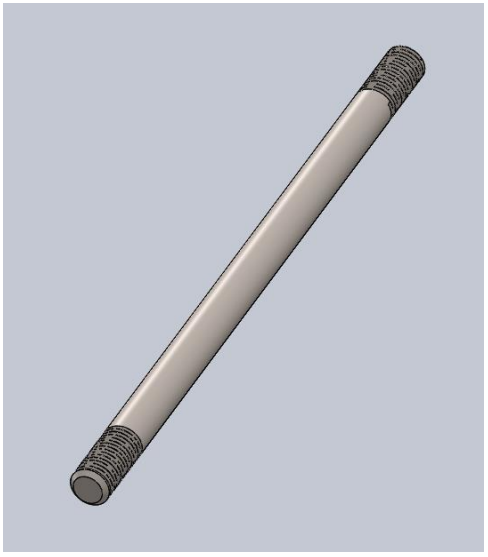


Figure 11: Z-Axis Linear Rod

2.2.1.9. Z-axis power screw

$$m = \rho V = 7500 \times \pi \times (8)^2 \times 100 \times 10^{-9} = 0.754\ Kg$$

Figure 12: Z-Axis Power screw

2.2.1.10. Z-axis motor holder

$$m = \rho V = 2710 \times (70 \times 45 \times 5) \times 10^{-9} = \ 0.043\ Kg$$

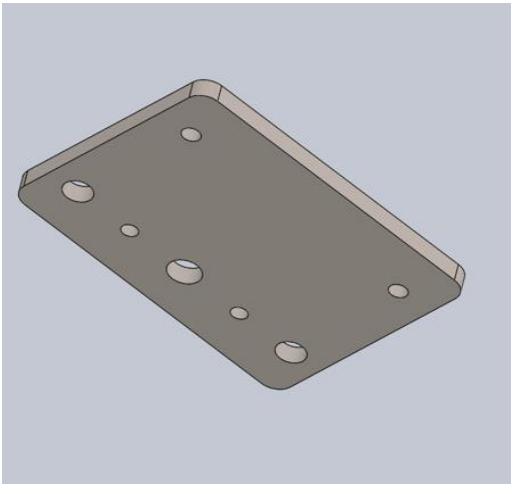


Figure 13: Z-Axis Motor Holder

2.2.1.11. Product

$m = \rho V = 1210 \times \pi \times (50)^2 \times 45 \times 10^{-9} = 0.4276\ Kg$

2.2.1.12. Standard Parts Masses

Bearing mass $m = 0.07\ Kg$

Linear bearing $m = 0.85\ kg$

2.2.2. X-axis lead screw calculations

Lead screw pitch $p = \frac{1}{2\times10^{-3}} = 500\ rev/mm$

Lead screw efficiency $\eta = 90\%$

$J_t\ddot{\theta} = T_m - T_R$

$J_t = J_m + J_L$

$J_L = \frac{m_L}{\eta(2\pi p)}$

$m_L = [2(0.17344) + 0.41192 + 0.245 + 0.754 + 2(0.13) + 0.096 + 2(0.082) + 0.151 + 2(0.043) + 2(0.2) + 4(0.07) + 3(0.85) + (0.4276)]/2 = 3.0862\ Kg$

$J_L = \frac{m_L}{\eta(2\pi p)^2} = \frac{3.0862}{0.9(2\pi \times 500)^2} = 3.474 \times 10^{-7}\ Kg.m^2$

$J_m = \frac{1}{10}J_L = \frac{1}{10} \times 3.474 \times 10^{-7} = 3.474 \times 10^{-8}\ Kg.m^2$

2.2.2.1. Torque resistance calculations

$F_L = \mu mg = 0.3 \times 3.0862 \times 9.81 = 9.083\ N$

$T_R = F.r = 9.083\ \times 4 \times 10^{-3} = 0.036\ N.m$

2.2.2.2. Velocity profile

$X = 0.31\ m$

$\theta_t = (2\pi p)X = 2\pi \times 500 \times 0.31 = 973.80\ rad$

$\theta_t = \int \dot{\theta} = Area\ under\ curve = \left[\frac{1}{2}t_a + t_c + \frac{1}{2}t_d\right]\dot{\theta}_{max} = \left[\frac{1}{2} \times 15 + 120 + \frac{1}{2} \times 15\right]\dot{\theta}_{max}$

$= 973.80\ rad$

$\dot{\theta}_{max} = 7.214\ rad/sec$

$\ddot{\theta} = \frac{d\dot{\theta}}{dt} = curve\ slope$

2.2.2.3. Acceleration

$\ddot{\theta}_a = \frac{7.214}{15} = 0.48\ rad/sec$

$J_t\ddot{\theta} = T_m - T_R$

$(3.474 \times 10^{-7} + 3.474 \times 10^{-8})0.48 = T_m - 0.036$

$T_m = 0.036\ N.m$

2.2.2.4. Constant Velocity

$\ddot{\theta}_a = 0\ rad/sec$

$J_t\ddot{\theta} = T_m - T_R$

$0 = T_m - 0.036$

$T_m = 0.036\ N.m$

2.2.2.5. Deacceleration

$\ddot{\theta}_d = \frac{-7.214}{15} = -0.48\ rad/sec$

$-(3.474 \times 10^{-7} + 3.474 \times 10^{-8})0.48 = T_m - 0.036$

$T_m = 0.036\ N.m$

$$T_{rms} = \sqrt{\frac{T_{ma}^2 t_a + T_{mc}^2 t_c + T_{md}^2 t_d}{t_a + t_c + t_d}} = \sqrt{\frac{(0.036)^2 \times 15 + (0.036)^2 \times 120 + (0.036)^2 \times 15}{15 + 120 + 15}}$$

$$= 0.036\ N.m$$

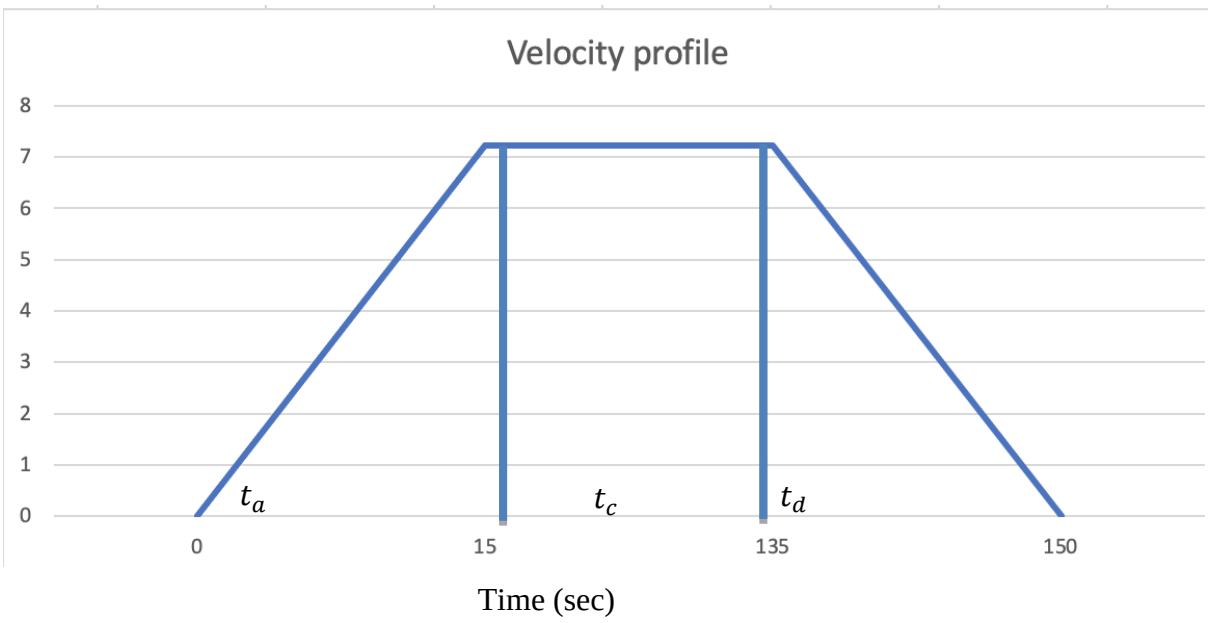


Figure 14: X-Axis Velocity Profile

2.2.3. Y-axis lead screw calculations

Lead screw pitch $p = \frac{1}{2\times10^{-3}} = 500\ rev/mm$

Lead screw efficiency $\eta = 90\%$

$J_t\ddot{\theta} = T_m - T_R$

$J_t = J_m + J_L$

$$J_L = \frac{m_L}{\eta(2\pi p)}$$

$$m_L = 2(0.13) + 0.0956 + 2(0.082) + 0.2 + 0.151 + 2(0.043) + 0.07 + 0.85 + 0.4276$$

$$= 2.1042\ Kg$$

$$J_L = \frac{m_L}{\eta(2\pi p)^2} = \frac{2.1042}{0.9(2\pi \times 500)^2} = 2.368 \times 10^{-7}\ Kg.m^2$$

$$J_m = \frac{1}{10}J_L = \frac{1}{10} \times 2.368 \times 10^{-7} = 2.368 \times 10^{-8}\ Kg.m^2$$

2.2.3.1. Torque resistance calculations

$F_L = \mu mg = 0.3 \times 2.1042 \times 9.81 = 6.193\ N$

$T_R = F.r = 6.193\ \times 4 \times 10^{-3} = 0.025\ N.m$

2.2.3.2. Velocity profile

$X = 0.28\ m$

$\theta_t = (2\pi p)X = 2\pi \times 500 \times 0.28 = 879.65\ rad$

$$\theta_t = \int \dot{\theta} = Area\ under\ curve = \left[\frac{1}{2}t_a + t_c + \frac{1}{2}t_d\right]\dot{\theta}_{max} = \left[\frac{1}{2} \times 15 + 120 + \frac{1}{2} \times 15\right]\dot{\theta}_{max}$$

$$= 879.65\ rad$$

$\dot{\theta}_{max} = 6.516\ rad/sec$

$$\ddot{\theta} = \frac{d\dot{\theta}}{dt} = curve\ slope$$

2.2.3.3. Acceleration

$$\ddot{\theta}_a = \frac{6.516}{15} = 0.434\ rad/sec$$

$J_t\ddot{\theta} = T_m - T_R$

$(2.368 \times 10^{-7}\ + 2.368 \times 10^{-8}\ )0.48 = T_m - 0.025$

$T_m = 0.025\ N.m$

2.2.3.4. Constant Velocity

$\ddot{\theta}_a = 0\ rad/sec$

$J_t\ddot{\theta} = T_m - T_R$

$0 = T_m - 0.025$

$T_m = 0.025\ N.m$

2.2.3.5. Deacceleration

$$\ddot{\theta}_d = \frac{-6.516}{15} = -0.48\ rad/sec$$

$-(2.368 \times 10^{-7} + 2.368 \times 10^{-8})0.48 = T_m - 0.025$

$T_m = 0.025\ N.m$

$$T_{rms} = \sqrt{\frac{T_{ma}^2 t_a + T_{mc}^2 t_c + T_{md}^2 t_d}{t_a + t_c + t_d}} = \sqrt{\frac{(0.025)^2 \times 15 + (0.025)^2 \times 120 + (0.025)^2 \times 15}{15 + 120 + 15}}$$

$$= 0.025\ N.m$$

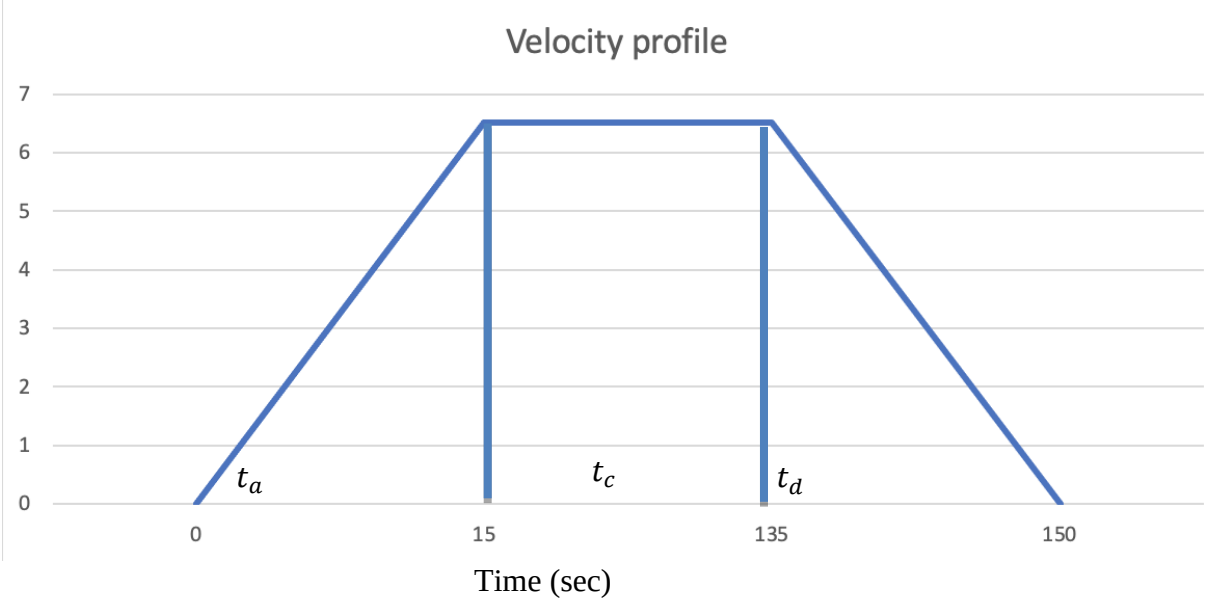


Figure 15: Y-Axis Velocity Profile

2.2.4. Z-axis lead screw calculations

Lead screw pitch $p = \frac{1}{2\times10^{-3}} = 500\ rev/mm$

Lead screw efficiency $\eta = 90\%$

$J_t\ddot{\theta} = T_m - T_R$

$J_t = J_m + J_L$

$$J_L = \frac{m_L}{\eta(2\pi p)}$$

$m_L = 0.043 + 2(0.085) + 0.4276 = 2.1706\ Kg$

$$J_L = \frac{m_L}{\eta(2\pi p)^2} = \frac{2.1706}{0.9(2\pi \times 500)^2} = 2.444 \times 10^{-7}\ Kg.m^2$$

$$J_m = \frac{1}{10}J_L = \frac{1}{10} \times 2.444 \times 10^{-7} = 2.444 \times 10^{-8}\ Kg.m^2$$

2.2.4.1. Torque resistance calculations

$F_L = \mu mg = 0.3 \times 2.1706 \times 9.81 = 6.388\ N$

$$T_R = F.r = 6.388\ \times 4 \times 10^{-3} = 0.026\ N.m$$

2.2.4.2. Velocity profile

$X = 0.065\ m$

$\theta_t = (2\pi p)X = 2\pi \times 500 \times 0.065 = 204.2\ rad$

$$\theta_t = \int \dot{\theta} = Area\ under\ curve = \left[\frac{1}{2}t_a + t_c + \frac{1}{2}t_d\right]\dot{\theta}_{max} = \left[\frac{1}{2}\times 5 + 20 + \frac{1}{2}\times 5\right]\dot{\theta}_{max}$$

$$= 204.2\ rad$$

$\dot{\theta}_{max} = 8.168\ rad/sec$

$$\ddot{\theta} = \frac{d\dot{\theta}}{dt} = curve\ slope$$

2.2.4.3. Acceleration

$$\ddot{\theta}_a = \frac{8.168}{5} = 0.27\ rad/sec$$

$J_t\ddot{\theta} = T_m - T_R$

$(2.444 \times 10^{-7} + 2.444 \times 10^{-8})0.48 = T_m - 0.026$

$T_m = 0.026\ N.m$

2.2.4.4. Constant Velocity

$\ddot{\theta}_a = 0\ rad/sec$

$J_t\ddot{\theta} = T_m - T_R$

$0 = T_m - 0.026$

$T_m = 0.026\ N.m$

2.2.4.5. Deacceleration

$$\ddot{\theta}_d = \frac{-8.168}{5} = -0.27\ rad/sec$$

$-(2.444 \times 10^{-7} + 2.44 \times 10^{-8})0.48 = T_m - 0.026$

$T_m = 0.026\ N.m$

$$T_{rms} = \sqrt{\frac{T_{ma}^2 t_a + T_{mc}^2 t_c + T_{md}^2 t_d}{t_a + t_c + t_d}} = \sqrt{\frac{(0.026)^2 \times 5 + (0.026)^2 \times 20 + (0.026)^2 \times 5}{5 + 20 + 5}}$$

$$= 0.026\ N.m$$

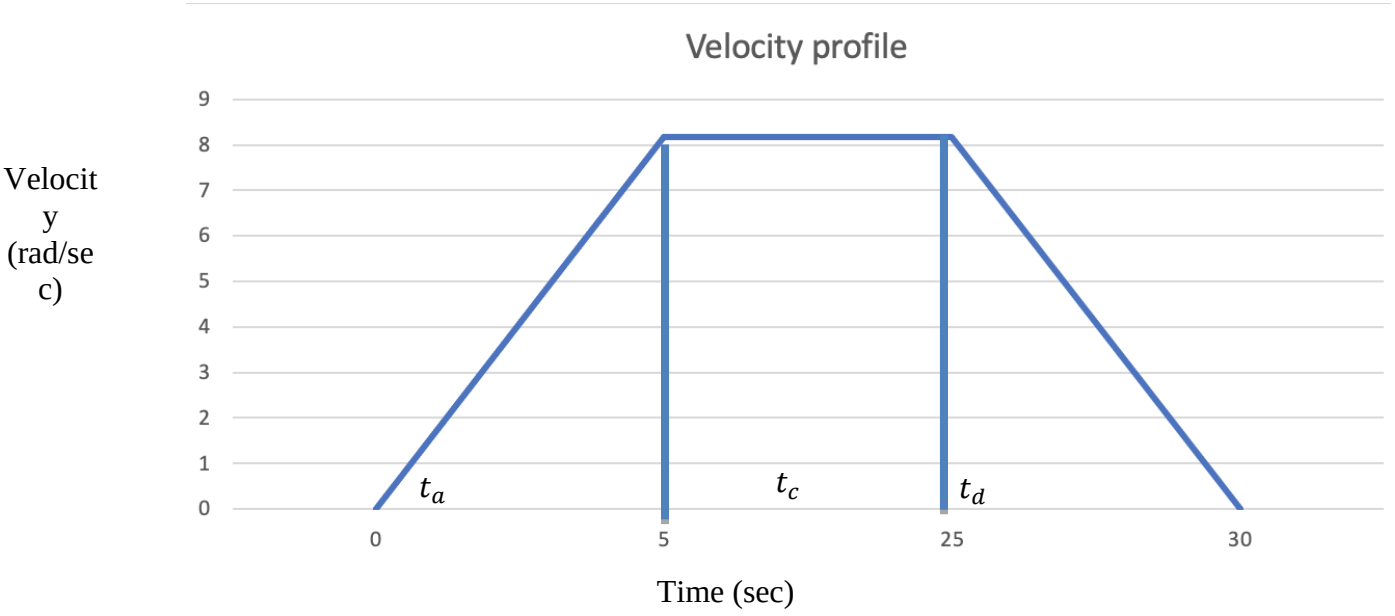


Figure 16: Z-Axis Velocity Profile

2.2.5. Nema 17 rated torque

Motor Properties	
Motor Type	Bipolar Stepper
Manufacturer Part Number	42BYGHM810
Step Angle	0.9°
Step Accuracy	5 %
Holding Torque	4.8 kg·cm
Rated Torque	4.3 kg·cm
Maximum Speed (w/1063 Motor Controller)	300 RPM
Maximum Motor Speed	2344 RPM
Acceleration at Max Speed (w/1067 Motor Controller)	2.4E+06 1/16 steps/sec²

$$T_{rated} = 4.8\ kg.cm = \ 0.4707192\ N.m > T_{rms}$$

2.2.6. Pneumatic Actuator Sizing

Friction Coefficient $\mu = 0.1$

Cylinder stroke $S = 0.1\ m$

2.2.6.1. Velocity profile

$t_a = 0.25\ sec$

$t_c = 1.5\ sec$

$t_d = 0.25\ sec$

$$S = \int Vdt = Area\ under\ curve = \left[\frac{1}{2}(0.25) + 1.5 + \frac{1}{2}(0.25)\right] V_{max} = 0.1\ m$$

$V_{max} = 0.057\ m/sec$

$$a = \frac{dv}{dt} = slope\ of\ curve$$

$$\sum F = ma$$

2.2.6.2. Acceleration

$$a_a = \frac{0.057}{0.25} = 0.229\ m/sec$$

$$F_C - F_{friction} = ma$$

$$F_{friction} = \mu mg$$

$$F_c - 0.1 \times 0.4276 \times 9.81 = 0.4276 \times 0.229$$

$$F_c = 0.517\ N$$

$$P = \frac{F}{A} = \frac{F}{\frac{\pi}{4} \times (D)^2} = \frac{0.517}{\frac{\pi}{4} \times (0.025)^2} = 1053.22\ Pa$$

2.2.6.3. Constant velocity

$a_a = 0\ m/sec$

$$F_C - F_{friction} = ma$$

$$F_{friction} = \mu mg$$

$$F_c - 0.1 \times 0.4276 \times 9.81 = 0$$

$$F_c = 0.419\ N$$

$$P = \frac{F}{A} = \frac{F}{\frac{\pi}{4} \times (D)^2} = \frac{0.419}{\frac{\pi}{4} \times (0.025)^2} = 853.58\ Pa$$

2.2.6.4. Deacceleration

$$a_a = -\frac{0.057}{0.25} = -0.229\ m/sec$$

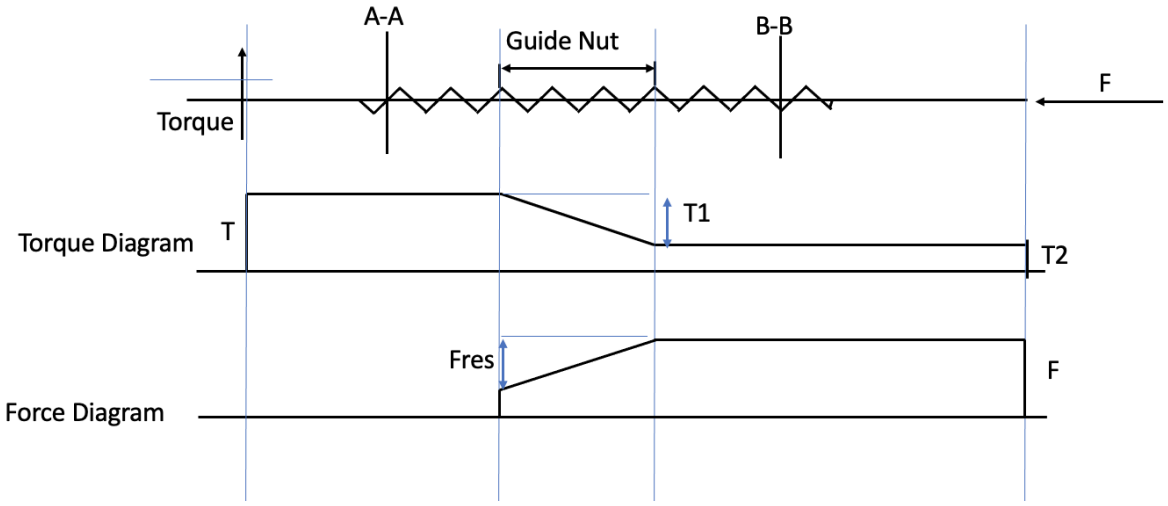
$$F_C - F_{friction} = ma$$

$$F_{friction} = \mu mg$$

$$F_c - 0.1 \times 0.4276 \times 9.81 = -0.4276 \times 0.229$$

$$F_c = 0.322\ N$$

$$P = \frac{F}{A} = \frac{F}{\frac{\pi}{4} \times (D)^2} = \frac{0.322}{\frac{\pi}{4} \times (0.025)^2} = 655.97\ Pa$$



2.3. Stress Calculations

2.3.1. X-axis Power Screw Calculations

Stainless steel (grade 37) trapezoidal power screw

$\sigma_y = 200\ MPa$

$\beta = 15°$

$P = 2\ mm$

$d_o = 8\ mm$

$d_r = d_o - P - 0.5 = 8 - 2 - 0.5 = 5.5\ mm$

$$d_m = \frac{d_o + d_r}{2} = \frac{8 + 5.5}{2} + 0.25 = 7\ mm$$

$$\tan(\alpha) = \frac{P}{\pi d_m} = \frac{2}{\pi(7)} = 0.909$$

$$\mu' = \frac{\mu}{\cos\beta} = \frac{0.2}{\cos(15)} = 0.2071$$

$F = 9.083\ N$

$T = T_1 + T_2$

$$T_1 = \frac{F d_m}{2}\left[\frac{\tan(\alpha) + \mu'}{1 - \mu' \tan(\alpha)}\right]$$

$$T_2 = \frac{F d_s}{2}\mu_s = 0$$

$$T_1 = \frac{9.093(7)}{2}\left[\frac{0.909 + 0.2071}{1 - 0.909(0.2071)}\right] = 43.71\ N.mm$$

$$\tau_{max} = \sqrt{(\tau)^2 + \left(\frac{\sigma}{2}\right)^2}$$

Section A-A

$$\tau_{max} = \tau = \frac{16T}{\pi d_r^3} = \frac{16(43.71)}{\pi(5.5)^3} = 1.34\ MPa$$

$$\tau_{max} \leq \frac{\sigma_y}{2n}$$

$$1.34 \leq \frac{200}{2n}$$

$n = 74.62$ → suitable

Section B-B

$T_2 = 0$

$F = 9.083\ N$

$$\sigma = \frac{F}{\frac{\pi}{4} d_r^3} = \frac{9.083}{\frac{\pi}{4}(5.5)^3} = 0.382\ MPa$$

$$\tau_{max} = \frac{\sigma}{2} = 0.191\ MPa$$

$$\tau_{max} \leq \frac{\sigma_y}{2n}$$

$$0.191 \leq \frac{200}{2n}$$

$n = 523.51$

2.3.2. Y-axis Power Screw Calculations

Stainless steel (grade 37) trapezoidal power screw

$\sigma_y = 200\ MPa$

$\beta = 15°$

$P = 2\ mm$

$d_o = 8\ mm$

$d_r = d_o - P - 0.5 = 8 - 2 - 0.5 = 5.5\ mm$

$$d_m = \frac{d_o + d_r}{2} = \frac{8 + 5.5}{2} + 0.25 = 7\ mm$$

$$\tan(\alpha) = \frac{P}{\pi d_m} = \frac{2}{\pi(7)} = 0.909$$

$$\mu' = \frac{\mu}{\cos\beta} = \frac{0.2}{\cos(15)} = 0.2071$$

$F = 6.193\ N$

$T = T_1 + T_2$

$$T_1 = \frac{F d_m}{2}\left[\frac{\tan(\alpha) + \mu'}{1 - \mu' \tan(\alpha)}\right]$$

$$T_2 = \frac{F d_s}{2}\mu_s = 0$$

$$T_1 = \frac{6.193(7)}{2}\left[\frac{0.909 + 0.2071}{1 - 0.909(0.2071)}\right] = 29.8\ N.mm$$

$$\tau_{max} = \sqrt{(\tau)^2 + \left(\frac{\sigma}{2}\right)^2}$$

Section A-A

$$\tau_{max} = \tau = \frac{16T}{\pi d_r^3} = \frac{16(29.8)}{\pi(5.5)^3} = 0.9122\ MPa$$

$$\tau_{max} \leq \frac{\sigma_y}{2n}$$

$$0.9122 \leq \frac{200}{2n}$$

$n = 109.63$ → suitable

Section B-B

$T_2 = 0$

$F = 6.193\ N$

$$\sigma = \frac{F}{\frac{\pi}{4} d_r^3} = \frac{6.193}{\frac{\pi}{4}(5.5)^3} = 0.261\ MPa$$

$$\tau_{max} = \frac{\sigma}{2} = 0.1305\ MPa$$

$$\tau_{max} \leq \frac{\sigma_y}{2n}$$

$$0.1305 \leq \frac{200}{2n}$$

$n = 766.28$

2.3.3. Z-axis Power Screw Calculations

Stainless steel (grade 37) trapezoidal power screw

$\sigma_y = 200\ MPa$

$\beta = 15°$

$P = 2\ mm$

$d_o = 8\ mm$

$d_r = d_o - P - 0.5 = 8 - 2 - 0.5 = 5.5\ mm$

$$d_m = \frac{d_o + d_r}{2} = \frac{8 + 5.5}{2} + 0.25 = 7\ mm$$

$$\tan(\alpha) = \frac{P}{\pi d_m} = \frac{2}{\pi(7)} = 0.909$$

$$\mu' = \frac{\mu}{\cos\beta} = \frac{0.2}{\cos(15)} = 0.2071$$

$F = 6.388\ N$

$T = T_1 + T_2$

$$T_1 = \frac{F d_m}{2}\left[\frac{\tan(\alpha) + \mu'}{1 - \mu' \tan(\alpha)}\right]$$

$$T_2 = \frac{F d_s}{2}\mu_s = 0$$

$$T_1 = \frac{6.388(7)}{2}\left[\frac{0.909 + 0.2071}{1 - 0.909(0.2071)}\right] = 30.74\ N.mm$$

$$\tau_{max} = \sqrt{(\tau)^2 + \left(\frac{\sigma}{2}\right)^2}$$

Section A-A

$$\tau_{max} = \tau = \frac{16T}{\pi d_r^3} = \frac{16(30.74)}{\pi(5.5)^3} = 0.941\ MPa$$

$$\tau_{max} \leq \frac{\sigma_y}{2n}$$

$$0.941 \leq \frac{200}{2n}$$

$n = 106.27$ → suitable

Section B-B

$T_2 = 0$

$F = 6.388\ N$

$$\sigma = \frac{F}{\frac{\pi}{4} d_r^3} = \frac{6.193}{\frac{\pi}{4}(5.5)^3} = 0.269\ MPa$$

$$\tau_{max} = \frac{\sigma}{2} = 0.1305\ MPa$$

$$\tau_{max} \leq \frac{\sigma_y}{2n}$$

$$0.1305 \leq \frac{200}{2n}$$

$n = 766.28$

2.3.4. X-axis Guide Nut Calculations

Brass Bearing pressure = 17 MPa

$P = 2\ mm$

$$N = \frac{H}{P} = \frac{20}{2} = 10$$

$d_o = 8\ mm$

$d_r = d_o - P - 0.5 = 5.5\ mm$

$$B.P = \frac{F}{\frac{\pi}{4}(d_o^2 - d_i^2)N} = \frac{9.083}{\frac{\pi}{4}((8)^2 - (5.5)^2)(10)} = 0.034\ MPa$$

$B,P < B,Pstd$ → suitable

2.3.5. Y-axis Guide Nut Calculations

Brass Bearing pressure = 17 MPa

$P = 2\ mm$

$$N = \frac{H}{P} = \frac{20}{2} = 10$$

$d_o = 8\ mm$

$d_r = d_o - P - 0.5 = 5.5\ mm$

$$B.P = \frac{F}{\frac{\pi}{4}(d_o^2 - d_i^2)N} = \frac{6.193}{\frac{\pi}{4}((8)^2 - (5.5)^2)(10)} = 0.0234\ MPa$$

$B,P < B,Pstd$ → suitable

2.3.6. Z-axis Guide Nut Calculations

Brass Bearing pressure = 17 MPa

$P = 2\ mm$

$N = \frac{H}{P} = \frac{20}{2} = 10$

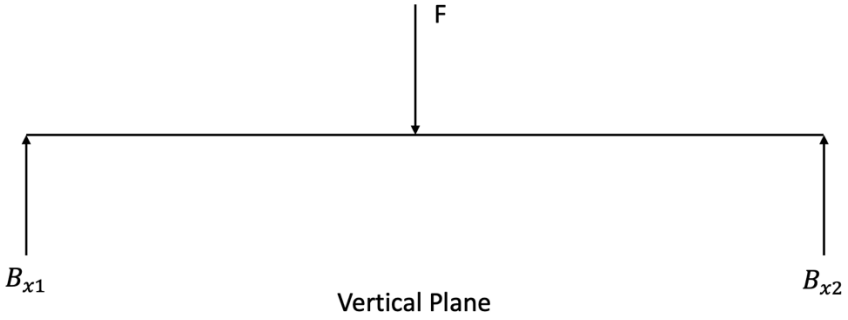
$d_o = 8\ mm$

$d_r = d_o - P - 0.5 = 5.5\ mm$

$B.P = \frac{F}{\frac{\pi}{4}(d_o^2 - d_i^2)N} = \frac{6.388}{\frac{\pi}{4}((8)^2 - (5.5)^2)(10)} = 0.0241\ MPa$

$B,P < B,Pstd$ → suitable

2.3.7. X-axis Bearing



$F = 9.083\ N$

$B_{x1} = B_{x2} = \frac{9.083}{2} = 4.5415\ N$

$B_{y1} = B_{y2} = 0$

$F_R = \sqrt{F_x^2 + F_y^2} = \sqrt{(4.5415)^2 + (0)^2} = 4.5415\ N$

$F_A = 0$

Bearing Type is self-aligning Ball Bearing

$C_o = 3.8\ KN$

$\frac{F_A}{F_R} = 0$

$\frac{F_A}{F_R} \leq e$

$X = 1$

$F_e = XVF_R + YF_A = (1)(1)4.5415 = 4.5415\ N$

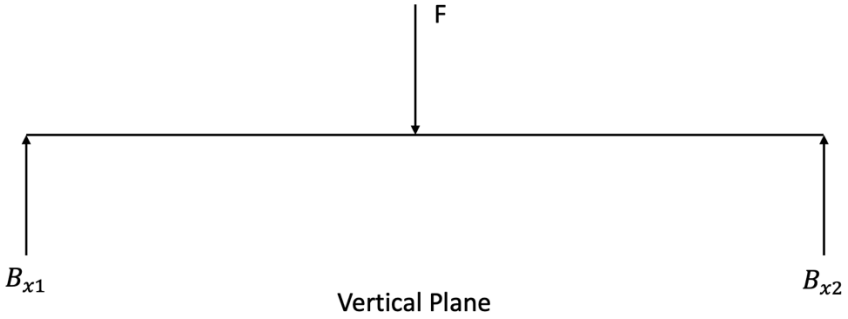
Assume the robot will work for 10000 hours

$B = \frac{L_h N 60}{10^6} = \frac{10000(10)(60)}{10^6} = 6\ rev\ per\ million\ life$

$C_{calc} = F_e \sqrt[3]{B} = 4.5415\sqrt[3]{6} = 8.252\ N$

$C_{calc} \leq C_o$ → suitable

2.3.8. Y-axis Bearing



$F = 6.193\ N$

$B_{x1} = B_{x2} = \frac{6.193}{2} = 3.0965\ N$

$B_{y1} = B_{y2} = 0$

$F_R = \sqrt{F_x^2 + F_y^2} = \sqrt{(3.0965)^2 + (0)^2} = 3.0965\ N$

$F_A = 0$

Bearing Type is self-aligning Ball Bearing

$C_o = 3.8\ KN$

$\frac{F_A}{F_R} = 0$

$\frac{F_A}{F_R} \leq e$

$X = 1$

$F_e = XVF_R + YF_A = (1)(1)3.0965 = 3.0965\ N$

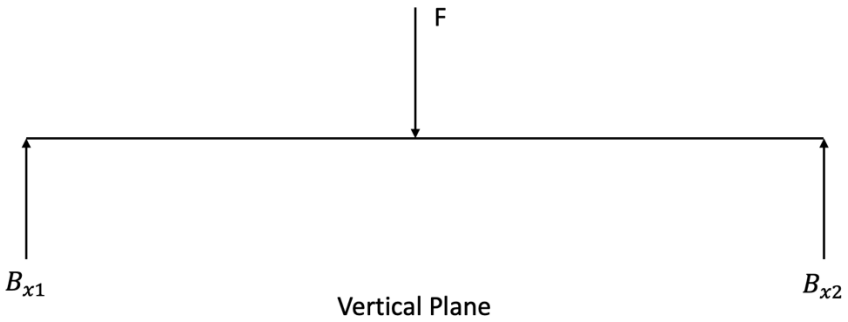
Assume the robot will work for 10000 hours

$B = \frac{L_h N 60}{10^6} = \frac{10000(10)(60)}{10^6} = 6\ rev\ per\ million\ life$

$C_{calc} = F_e \sqrt[3]{B} = 3.0965 \sqrt[3]{6} = 5.627\ N$

$C_{calc} \leq C_o \rightarrow$ suitable

2.3.9. Z-axis Bearing



$F = 6.388N$

$B_{x1} = B_{x2} = \frac{6.388}{2} = 3.194\ N$

$B_{y1} = B_{y2} = 0$

$F_R = \sqrt{F_x^2 + F_y^2} = \sqrt{(3.194)^2 + (0)^2} = 3.194\ N$

$F_A = 0$

Bearing Type is self-aligning Ball Bearing

$C_o = 3.8\ KN$

$\frac{F_A}{F_R} = 0$

$\frac{F_A}{F_R} \leq e$

$X = 1$

$F_e = XVF_R + YF_A = (1)(1)3.194 = 3.194\ N$

Assume the robot will work for 10000 hours

$B = \frac{L_h N 60}{10^6} = \frac{10000(10)(60)}{10^6} = 6\ rev\ per\ million\ life$

$C_{calc} = F_e \sqrt[3]{B} = 3.194 \sqrt[3]{6} = 5.804\ N$

$C_{calc} \leq C_o \rightarrow$ suitable

Bearing Type			In Relation to the Load the Inner Ring is		Single Row Bearings 1)		Double Row Bearings 2)				e
			Rotating	Stationary	$\frac{F_a}{VF_r} > e$		$\frac{F_a}{VF_r} \leqq e$		$\frac{F_a}{VF_r} > e$		
			V	V	X	Y	X	Y	X	Y	
3) Radial Contact Groove Ball Bearings	4) $\frac{F_a}{C_0}$	5) $\frac{F_a}{i Z D_w^2}$									
	0.014	25	1	1.2	0.56	2.30	1	0	0.56	2.30	0.19
	0.028	50				1.99				1.99	0.22
	0.056	100				1.71				1.71	0.26
	0.084	150				1.55				1.55	0.28
	0.11	200				1.45				1.45	0.30
	0.17	300				1.31				1.31	0.34
	0.28	500				1.15				1.15	0.38
	0.42	750				1.04				1.04	0.42
	0.56	1000				1.00				1.00	0.44
20°			1	1.2	0.43	1.00	1	1.09	0.70	1.63	0.57
25°					0.41	0.87		0.92	0.67	1.44	0.68
30°					0.39	0.76		0.78	0.63	1.24	0.80
35°					0.37	0.66		0.66	0.60	1.07	0.95
40°					0.35	0.57		0.55	0.57	0.93	1.14
Self-Aligning Ball Bearings			1	1	0.40	0.4 cot α	1	0.42 cot α	0.65	0.65 cot α	1.5 tan α
Self-Aligning and Tapered Roller Bearings			1	1.2	0.40	0.4 cot α	1	0.45 cot α	0.67	0.67 cot α	1.5 tan α

1) For single row bearings, when $\frac{F_a}{VF_r} \leqq e$ use $X = 1$ and $Y = 0$.
For two single row angular contact ball or roller bearings mounted "face-to-face" or "back-to-back" the values of X and Y which apply to double row bearings. For two or more single row bearings mounted "in tandem" use the values of X and Y which apply to single row bearings.
2) Double row bearings are presumed to be symmetrical.
3) Permissible maximum value of $\frac{F_a}{C_0}$ depends on the bearing design.
4) C_0 is the basic static load rating.
5) Units are pounds and inches.
Values of X, Y and e for a load or contact angle other than shown in the table are obtained by linear interpolation.

43

Figure 17: Bearing Types

Radial Bearings

Dimension-group 2 — Width-group 2 (Light Series, Wide Type)

Bore Reference Number	Dimensions mm			Bearing Series													
	Bearing Bore	Outside Diameter	Width	32[1])			22 22..K[2])		NU 22 (WUL,[3]) NJ 22 (WJL) NUP 22 (WUPL)		322[1])			222..HL 222..K HL[2])		
	d	D	b	b_1	C	C_0	C	C_0	C	C_0	B	C	C_0	C	y	C_0
00	10	30	14	14.0	695	430	490	200								
01	12	32	14	15.9	780	530	530	220								
02	15	35	14	15.9	780	655	585	236								
03	17	40	16	17.5	1.1	780	850	310								
04	20	47	18	20.6	1.53	1.1	1.0	430								
05	25	52	18	20.6	1.73	1.32	1.06	455	1.66	1.32				2.45	3.1	2.16
06	30	62	20	23.8	2.5	1.96	1.37	600	2.32	1.9	21.5	3.25	3.0	3.1	3.5	2.9
07	35	72	23	27	3.35	2.75	1.93	880	3.55	2.85	24.5	4.3	4.0	3.8	3.6	3.6
08	40	80	23	30.2	3.8	3.2	2.12	980	4.15	3.45	25	4.8	4.5	4.4	3.9	4.4
09	45	85	23	30.2	4.25	3.75	2.32	1.1	4.4	3.75	25	5.2	5.1	4.5	4.3	4.5
10	50	90	23	30.2	4.75	4.0	2.4	1.14	4.55	4.05	25	5.3	5.3	5.0	4.6	5.0
11	55	100	25	33.3	5.4	4.9	2.75	1.34	5.4	4.75	27	6.05	6.8	6.2	4.7	6.4
12	60	110	28	36.5	6.55	5.85	3.45	1.73	7.1	6.2	30	8.3	8.15	7.65	4.6	7.5
13	65	120	31	38.1	7.1	6.95	4.3	2.2	8.5	7.5	33	10.0	10.0	9.05	4.5	9.5
14	70	125	31	39.7	7.1	7.8	4.5	2.32	9.0	8.0	33.5	10.2	10.0	10.4	4.7	10.8
15	75	130	31	41.3	7.8	8.3	4.75	2.45	9.65	8.65	33.5	10.8	10.8	11.0	4.9	11.0
16	80	140	33	44.4	9.5	9.3	5.2	2.85	11.4	10.2	35.5	12.5	12.5	12.2	4.9	12.7
17	85	150	36	49.2	10.2	11.2	6.1	3.2	13.2	11.8	39	14.3	14.6	15.3	4.8	16.0
18	90	160	40	52.4	11.8	12.5	7.1	3.9	15.3	13.4	43	17.3	17.3	18.6	4.5	19.0
19	95	170	43	55.6	13.7	14.6	8.3	4.65	18.3	16.3	46	19.6	19.6	21.6	4.5	22.4
20	100	180	46	60.3	14.6	16.6			20.4	18.6	49.5	22.0	22.4	25.0	4.5	25.0

Figure 18: Bearing Types 2

4. ASSEMBLED LINE

Figure 19: Full Assembled Production Line

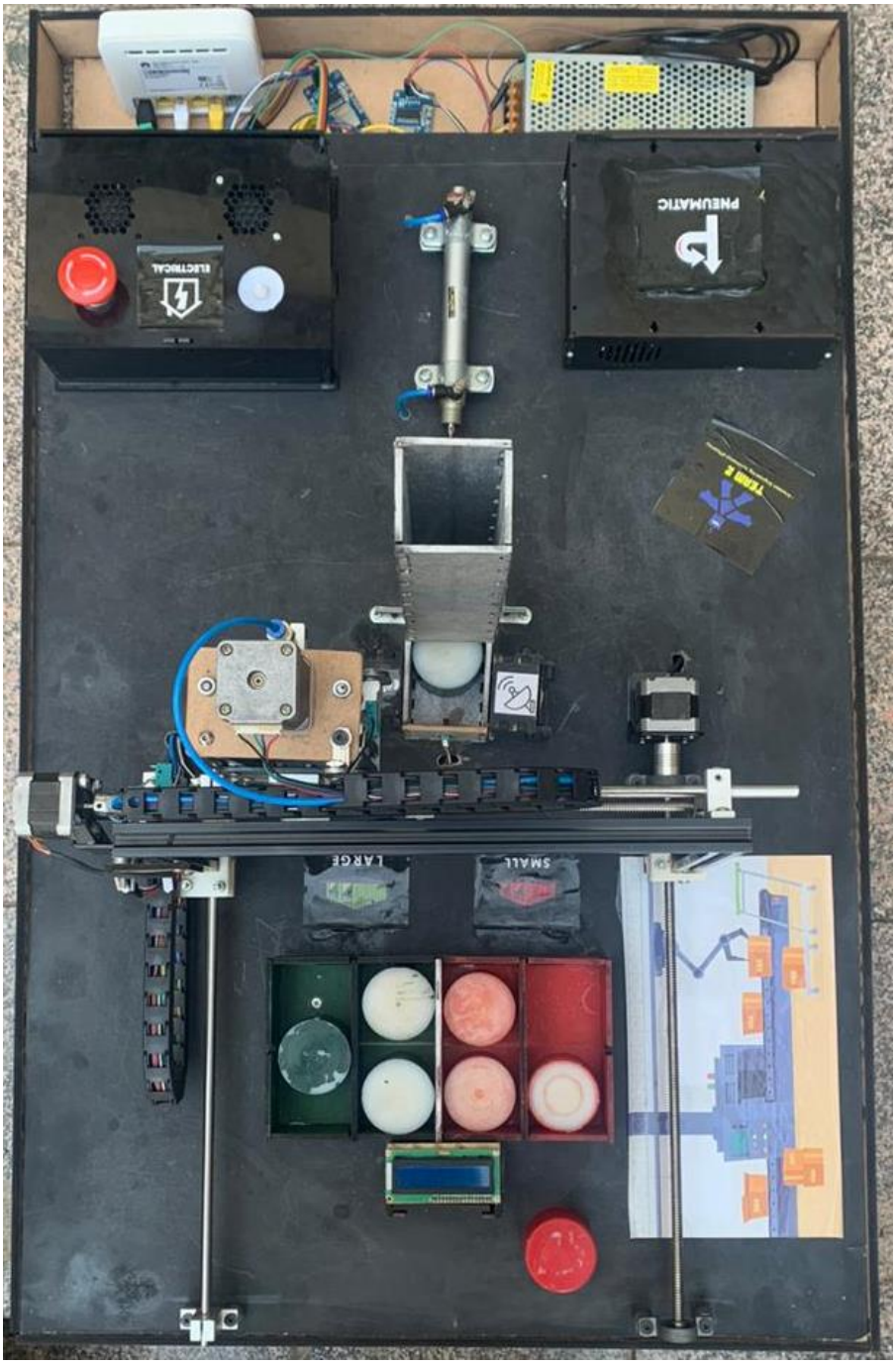


Figure 20: Production Line Top View

Figure 21: Production Line Reverse Isometric View

4.1. Feeding Terminal

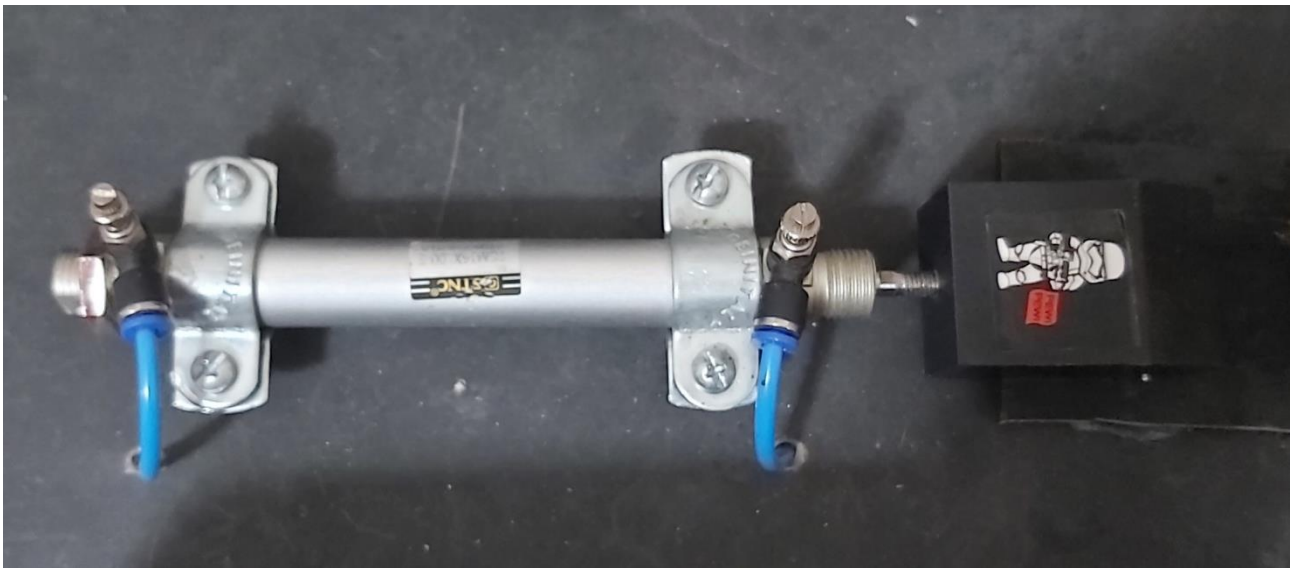


Figure 22: Feeding Terminal

Using the famous 2-way pneumatic cylinder together with 5/2 DCV (Directional Control Valve), fig 21, to control the extension and retraction of the piston rod assembled with a 3D Printed Punch that satisfies our needs:

- Push the product waiting in the magazine to be fed
- Supports the rest of products from falling in the magazine to allow timing between products feeding

Figure 23: 5/2 Solenoid Valve

For the feeding process to be done, we first need to check for product availability at the magazine so here comes the role of limit switches

Figure 24: Product Availabilty Checking Limit Switch

As shown in figure 22, a limit switch is available at the bottom of the magazine so when pressed the controller knows a product is available and so when the feeding time comes the piston cylinder will operate, else no feeding will occur.

Figure 25: Feeding Feedback Limit Switch

Figure 23 shows a limit switch located at the end of the magazine which functions as a feedback to the controller that a product reached the end position of the magazine and so the controller retracts the piston cylinder.

4.2. Sorting Terminal

Figure 26: Laser Transmitter Casing

Laser Transmitter from one side of the magazine emits laser beam which is received by an infrared receiver from the other side so that when the laser beam is cut then controller identifies the product as the large one, else it's the small one.

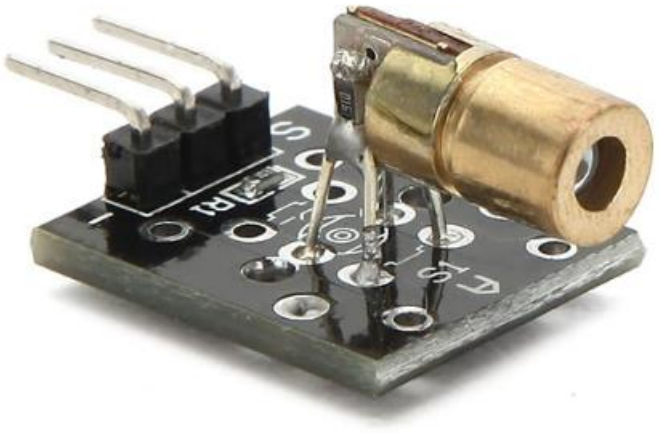


Figure 27: Laser Transmitter Module

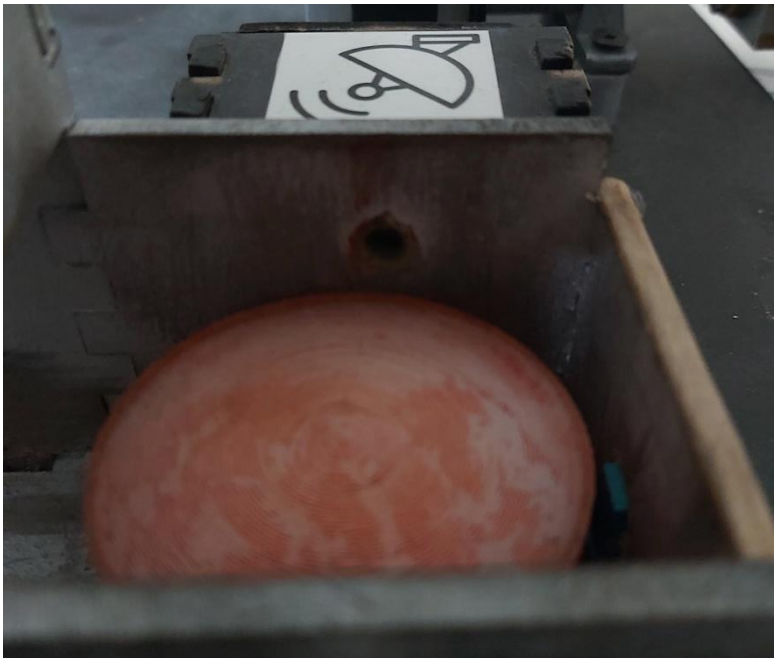


Figure 28: Laser Receiver Casing

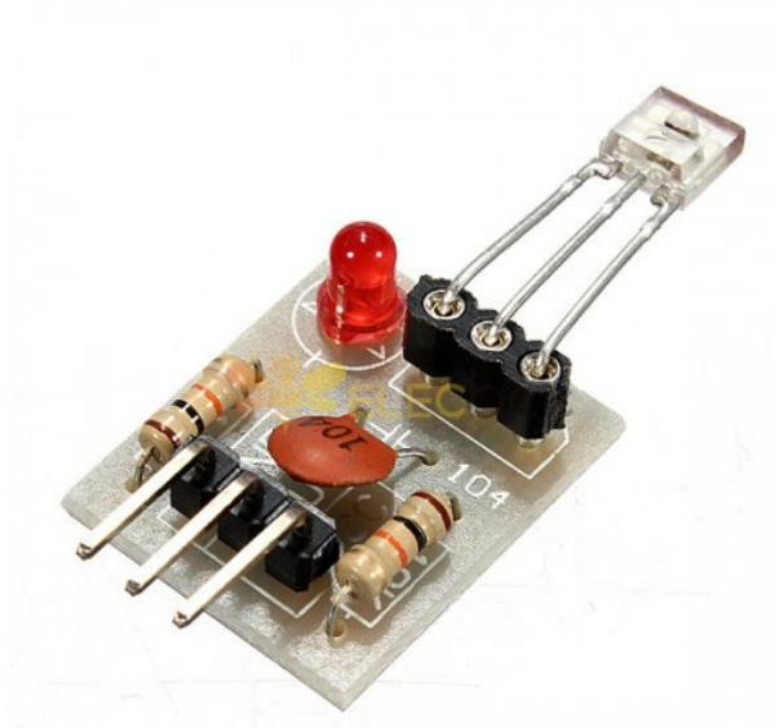


Figure 29: Laser Receiver Module

4.3. Robotic Arm

4.3.1. X-Axis

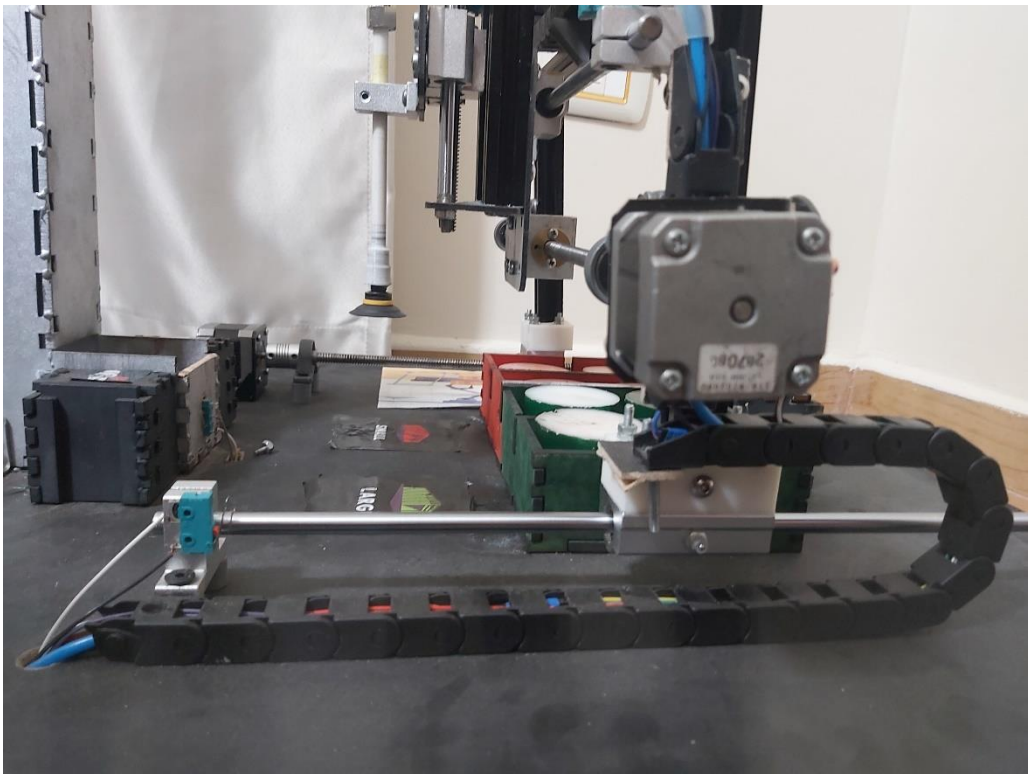


Figure 30: Robotic Arm X-Axis

X-axis of the robotic arm consists of a Nema-17 stepper motor rotating a lead screw supported by bearings from 2 the 2 sides, and a linear shaft that is supported with shafts fixer. The X-axis movement comes from the rigid linkage between the guide nut housing on the lead screw and the linear bearing on the linear shaft which allows a smooth movement along the axis of the lead screw. Homing of the X-axis is achieved using the limit switch shown illustrated in figure 28. Owing to the Towline, wirings are easily bent and extended along the full stroke of the axis.

4.3.2. Y-Axis

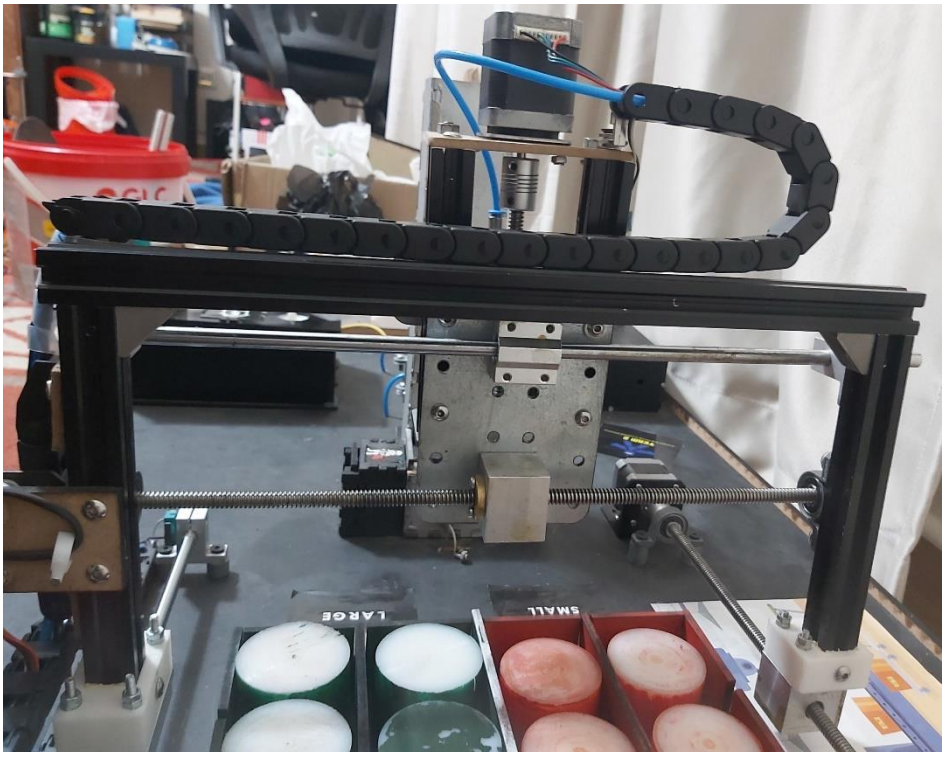


Figure 31: Robotic Arm Y-Axis

Same as the X-Axis, the Y-axis is based on the linkage between guide nut housing connected to lead screw and a linear bearing on linear shaft. Steel plate is fixed to the linear bearing and the guide nut housing firstly to mate the movement of the guide nut with the linear bearing and secondly to provide a fixation location for the Z-Axis components.

4.3.3. Z-Axis

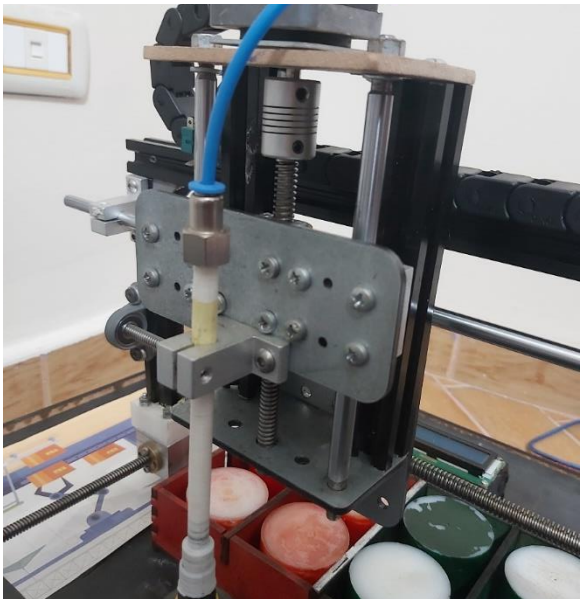


Figure 32: Robotic Arm Z-Axis

The new feature introduced in this axis is the integration between the linear movement of the components fixed with the guide nut housing and the linear bearing, and the suction feature that is used to keep holding the product while moving in the 3-axis space to store the product.

4.4. Storage

Figure 33: Storage

Storage unit consists of 2 products categories location with 4 locations for each category. The storage unit is accessorized with an LCD that shows the count stored for each category versus the other one.

4.5. Electrical Control Box

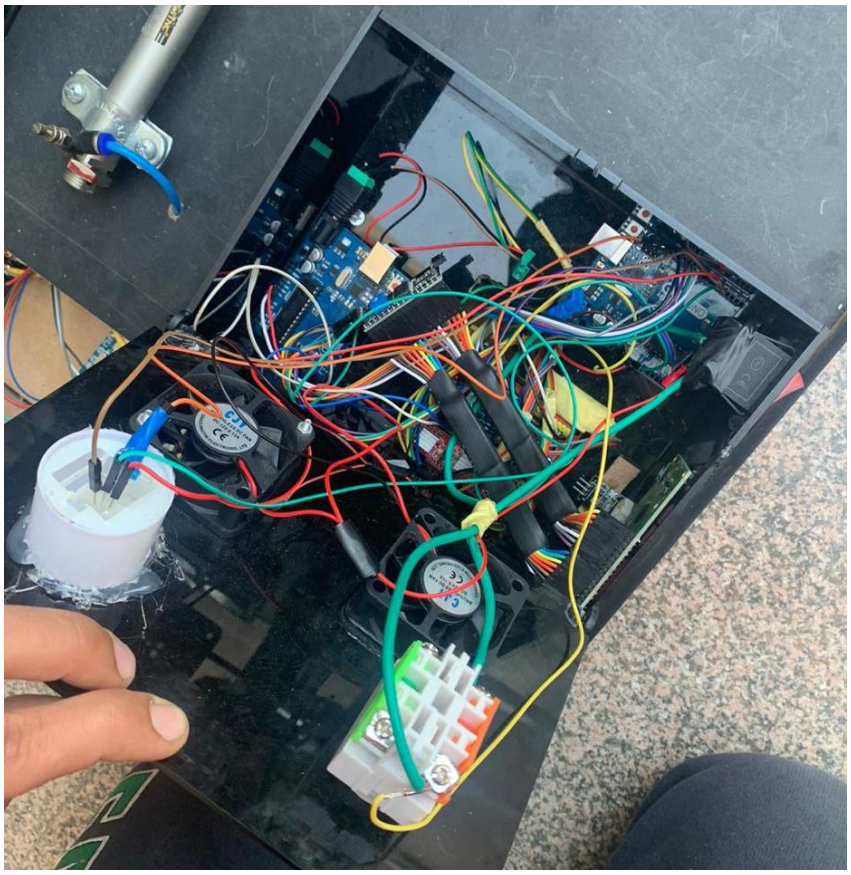


Figure 34: Electrical Control Box

Electrical Control Box contains the microcontrollers that control all the terminals of the production line, the control box is suited with 2 power switches one for powering up the production line and the other one for the cooling feature, which consists of 2 12V cooling fans to avoid overheating of the electrical components. Also, there's a RGB led fitted in semi-transparent cover to notifies about the state of operation of the production line. An LCD that displays the momentarily functioning process is provided in the control box. Emergency button is provided for cutting the power to the whole system in case malfunctioning or danger situations. For ethernet, SPI to ethernet modules are available under the control box together with the router. Last but not least, an IR receiver is provided to be able to start, hold, resume, and abort the system at any time using a remote control.

4.5.1. Arduino Uno

Figure 35: Arduino Uno

Arduino UNO is a microcontroller board based on the **ATmega328P**. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, and a 16 MHz ceramic resonator.

4.5.2. Arduino Nano

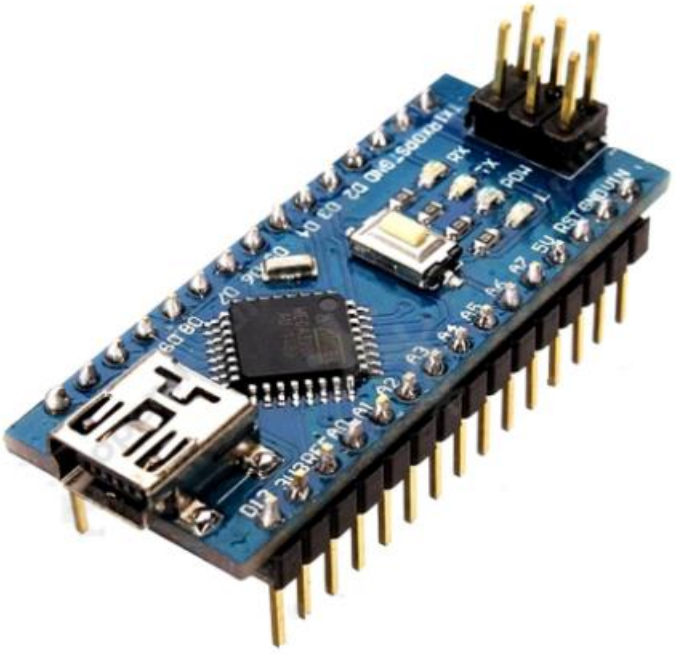


Figure 36: Arduino Nano

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328P released

in 2008. It offers the same connectivity and specs of the Arduino Uno board in a smaller form factor.[1]

The Arduino Nano is equipped with 30 male I/O headers, in a DIP-30-like configuration, which can be programmed using the Arduino Software integrated development environment (IDE), which is common to all Arduino boards and running both online and offline. The board can be powered through a type-B mini-USB cable or from a 9 V battery.

4.5.3. Arduino Nano CNC Shield

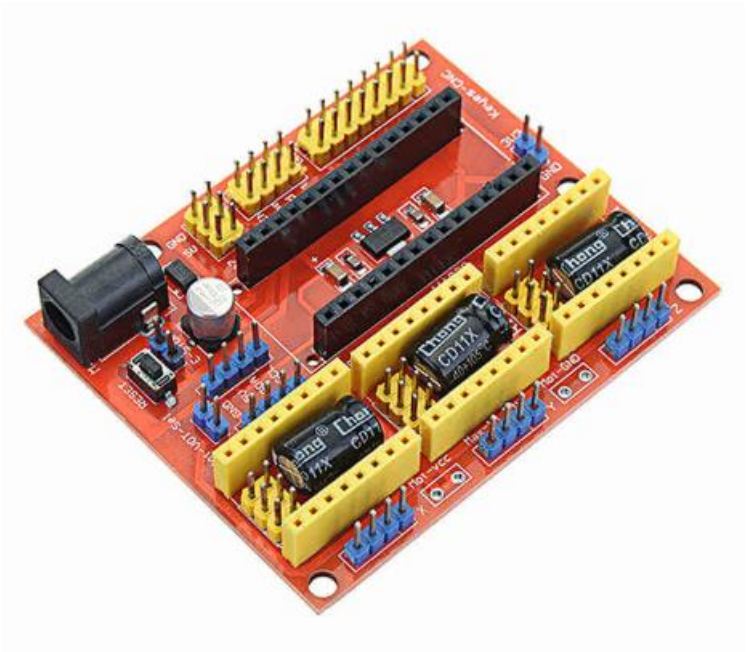


Figure 37: Arduino Nano CNC Shield

3D printer stepper motor driver CNC shield V4 needs to work with Nano board. It can be used as driver expansion board for engraving machines and 3D printers. It has in total 3 channel slots for A4988 stepper motor driver modules for driving 3 channel of stepper motors. Each channel of stepper motor only needs 2 IO ports, which means 6 IO ports is sufficient to manage 3 stepper motors. Furthermore, it provides connecting pins for limit switches that apply to all 3 axes. Provides connectivity for Serial communication using UART and provides another connecting port for I2C communication.

4.5.4. A4988 Stepper Motor Driver

Figure 38: A4988 Stepper Motor Driver With Heat Sink

Due to the simplicity of the step motor control and the variety of stepping modes provided by the A4988 driver, it is an ideal solution for building applications that require precise and reliable stepper motor control. The fact that it only requires two pins to control the speed and direction of a bipolar stepper motor like the NEMA 17 is pretty neat, too. Furthermore, the output current is regulated, allowing for noiseless operation of the stepper motor and the elimination of resonance, or ringing that is common in unregulated stepper driver designs.

The driver has a built-in translator for easy operation. This reduces the number of control pins to just two, one for controlling the steps and the other for controlling the spinning direction.

The driver offers five different step resolutions: full-step, half-step, quarter-step, eighth-step, and sixteenth-step.

4.5.5. LCD

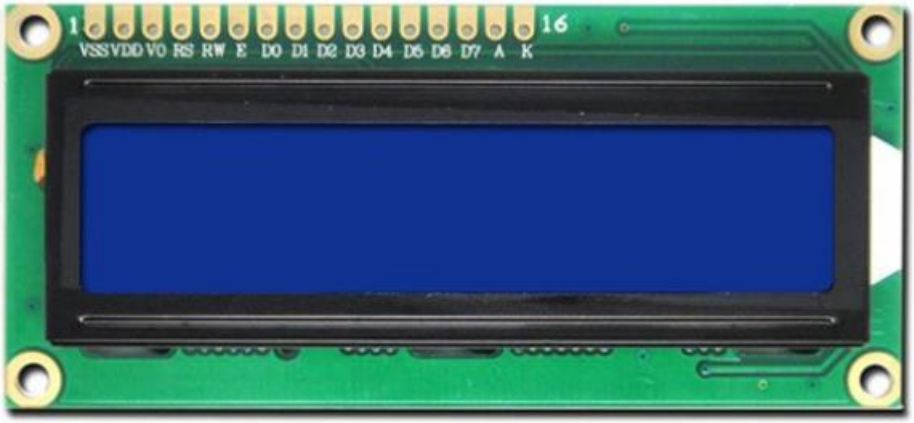


Figure 39: LCD (Liquid Crystal Display)

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

- A **register select (RS) pin** that controls where in the LCD's memory we're writing data to.
- A **Read/Write (R/W) pin** that selects reading mode or writing mode
- An **Enable pin** that enables writing to the registers
- 8 **data pins (D0 -D7)**. The states of these pins (high or low) are the bits that we're writing to a register when we write, or the values we're reading when we read.

There's also a display contrast pin (Vo), power supply pins (+5V and GND) and LED Backlight pins that can be used to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

4.5.6. I2C Module

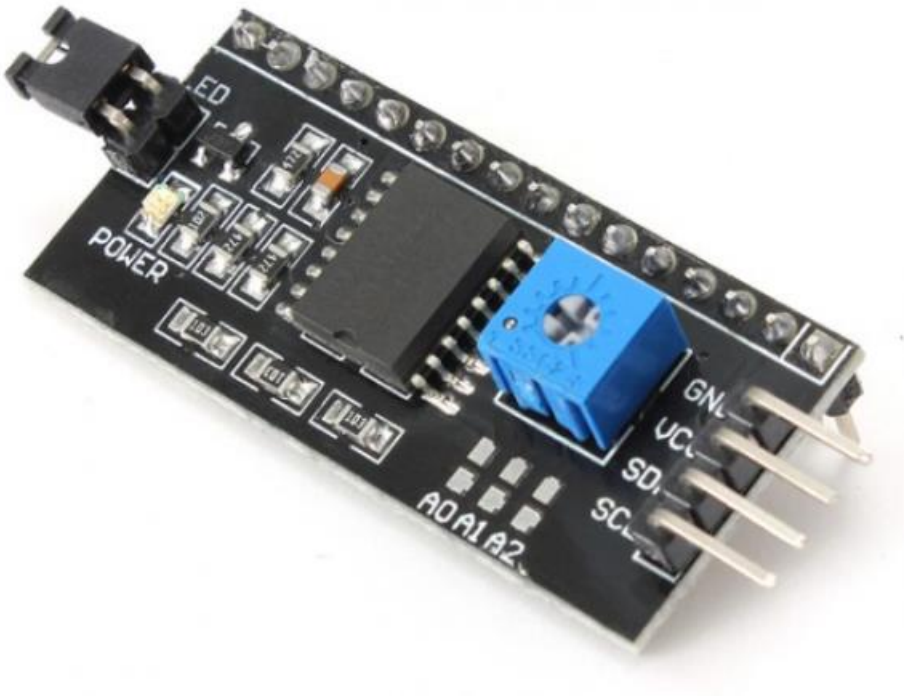


Figure 40: I2C Serial Communication Module

This module makes LCD connection easier by consuming only 2 pins from the Arduino DIO (Digital Input Output Pin) instead on 7-11 pins using the normal LCD. At the heart of the adapter is an 8-bit I/O expander chip – PCF8574. This chip converts the I2C data from an Arduino into the parallel data required for an LCD display. The board also comes with a small trim pot to make fine adjustments to the display's contrast. In addition, there is a jumper on the board that supplies power to the backlight. To control the intensity of the backlight, we can remove the jumper and apply external voltage to the header pin that is marked 'LED'.

4.5.7. IR Receiver Module

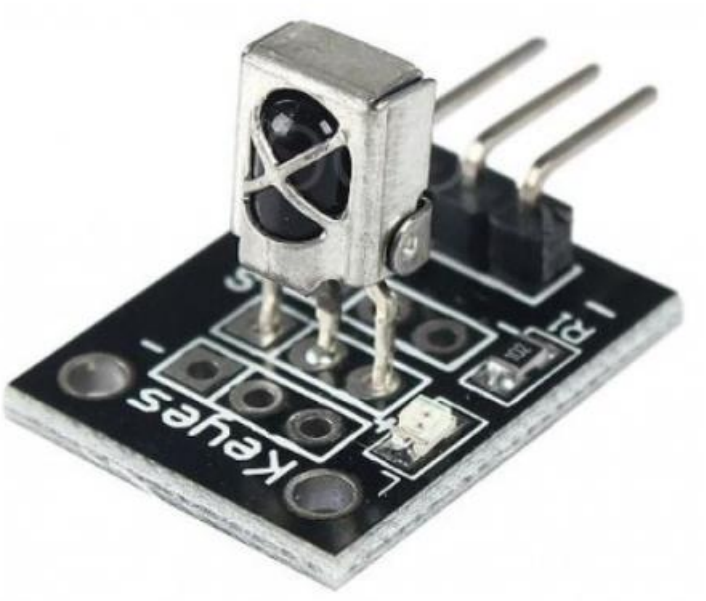


Figure 41: IR Receiver Module

The Infrared Receiver is used to receive infrared signals and also used for remote control detection. There is a IR detector on the Infrared Receiver which is used to get the infrared light emitted by the Infrared Emitter.

4.5.8. Power Switch

Figure 42: Power Switch

Used to power On/Off the production line and to turn the cooling fans On or Off. This switch has 3 legs, but we just connect 2 of them the middle one connect to the accessories we are connecting, while the legs at the '0' side of the switch is connected to the external power source.

4.5.9. Emergency Switch

Figure 43: Emergency Switch

Normally open switch that is used to break the passing through voltage by turning the circuit to an open one by pressing the red cap. Ideal for emergency and danger situations.

4.5.10. 12V Cooling Fan

Figure 44: 12V Cooling Fan

To avoid overheating of the electronics available at the control box, 2 fans are provided to never mind about heating.

4.5.11. Connectors

Figure 45: 0.5mm Cable

To withstand the high current coming out of the power supply we used the 0.5mm cables

Figure 46: Jumpers

For sensors and peripherals connection, thin, light jumpers are ideal.

Figure 47: Ethernet Cables

Used for the ethernet connection between microcontrollers

4.6. Pneumatic Control Box

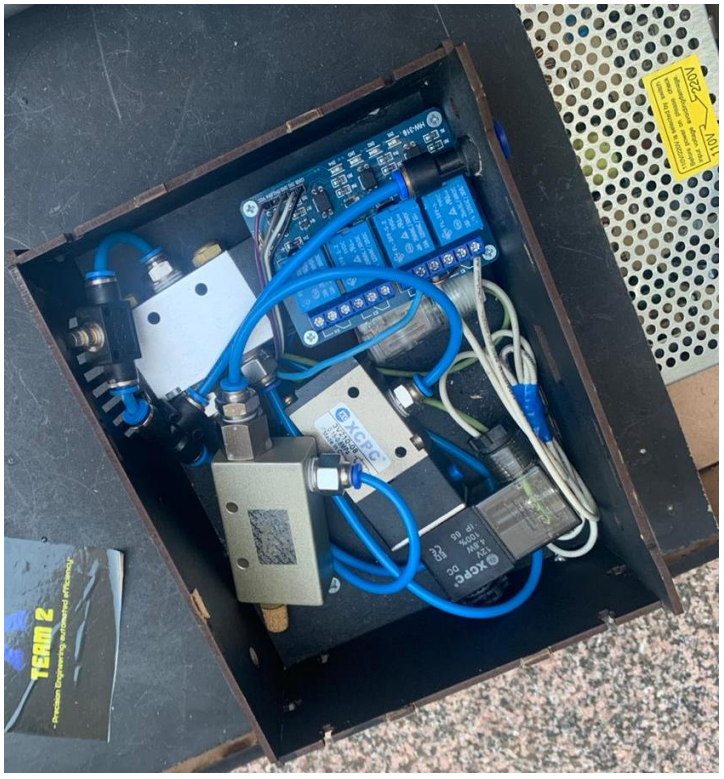


Figure 48: Pneumatic Control Box

The pneumatic control box is where we will find all the components related to the feeding and suction mechanisms.

4.6.1. 3/2 Solenoid Valve

Figure 49: 3/2 Solenoid Valve

Using the 3/2 solenoid valve in figure 31 the suction process was controlled. By opening air flow through the valve, the air goes to the vacuum generator in figure 32 which generates a vacuum between the suction cup contact circumference and the product itself which is then seen as suction process.

4.6.2. Vacuum Generator

Figure 50: Vacuum Generator

4.6.3. Vacuum Generator Working Principle

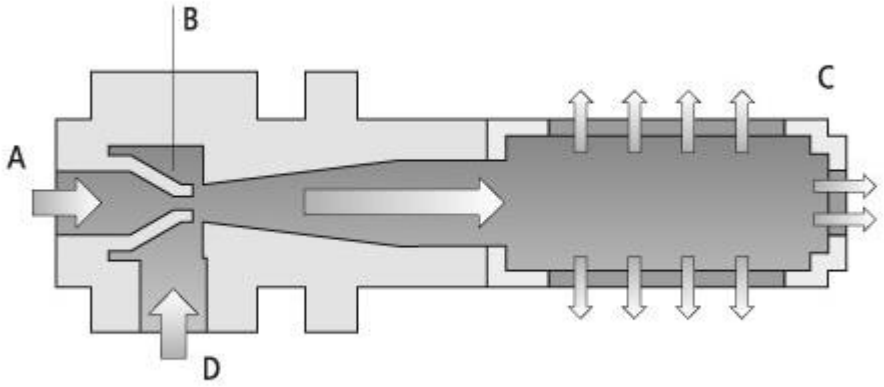


Figure 51: Vacuum Generator Internally

- Pneumatic vacuum generator's function based on the Venturi principle
- Compressed air is introduced into the ejector (A)
- Due to the reduced cross-section of the motive nozzle (the Venturi nozzle [B]), the compressed air is accelerated. The dynamic pressure increases, while the static air pressure simultaneously decreases.
- Once it has passed the motive nozzle, the accelerated air expands, and a vacuum is generated
- Air is "sucked" through the vacuum connection (D) into the ejector
- The compressed air escapes from the ejector through the silencer (C) together with the "sucked-in" air.

4.6.4. 5/2 Solenoid Valve

Figure 52: 5/2 Solenoid Valve

One input, 2 outputs, and 2 exhaust ports are provided by this valve, of which the input port comes from the compressor and the 2 outputs are connected to the pneumatic piston cylinder 1 for extension and the other for retraction.

4.6.5. Air Flow Control Valve

Figure 53: Air Flow Control Valve

Flow control valves are used in pneumatic systems to regulate the flow rate of compressed air. By controlling the flow rate, the speed of the pneumatic cylinder can also be regulated directly. In addition, a good throttling valve contributes to reducing wear due to a lower kinetic load.

4.6.6. Fittings

Figure 54: Tee Connector

Figure 55: 6-4 Fitting

Figure 56: Male Connector

Fittings are used to connect between pneumatic components and each other.

4.7. Extra Components

4.7.1. ENC28J60 SPI Ethernet Module

Figure 57: ENC28J60 SPI Ethernet Module

4.7.2. Router

Figure 58: Router

4.7.3. 12V 10A Power Supply

Figure 59: 12V 10A Power Supply

3. RTOS Flowchart

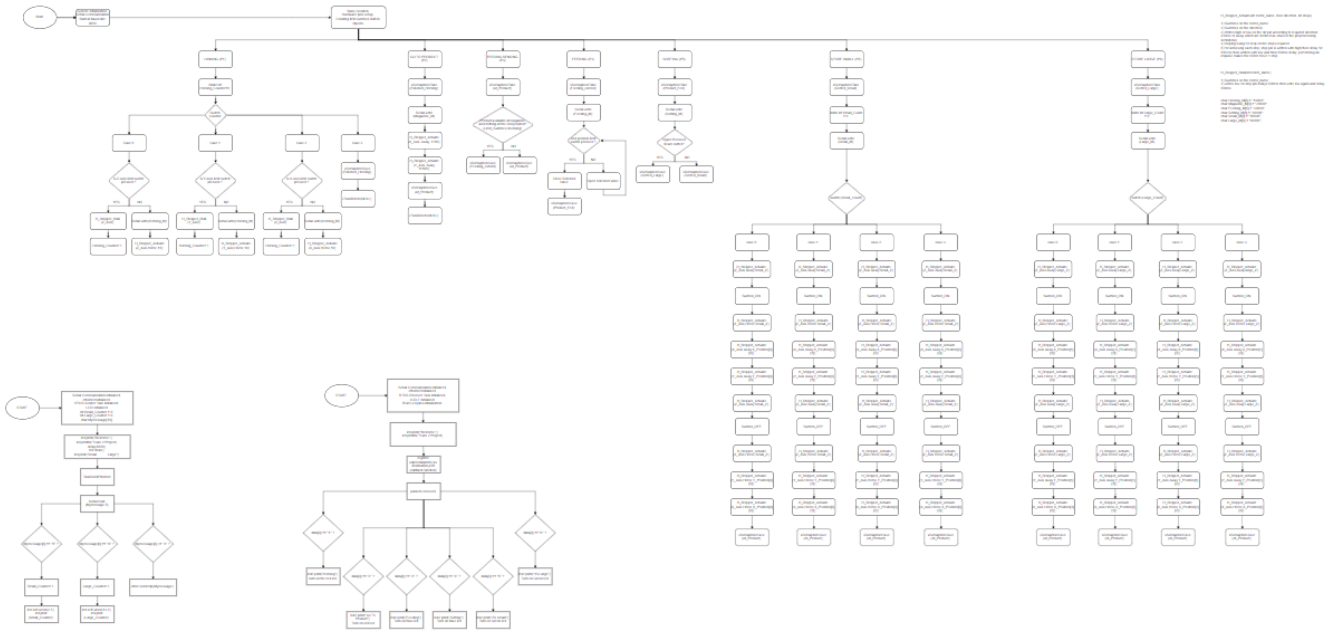


Figure 60: Full RTOS Flowchart

3.1. First MCU (Micro-Controller Unit)

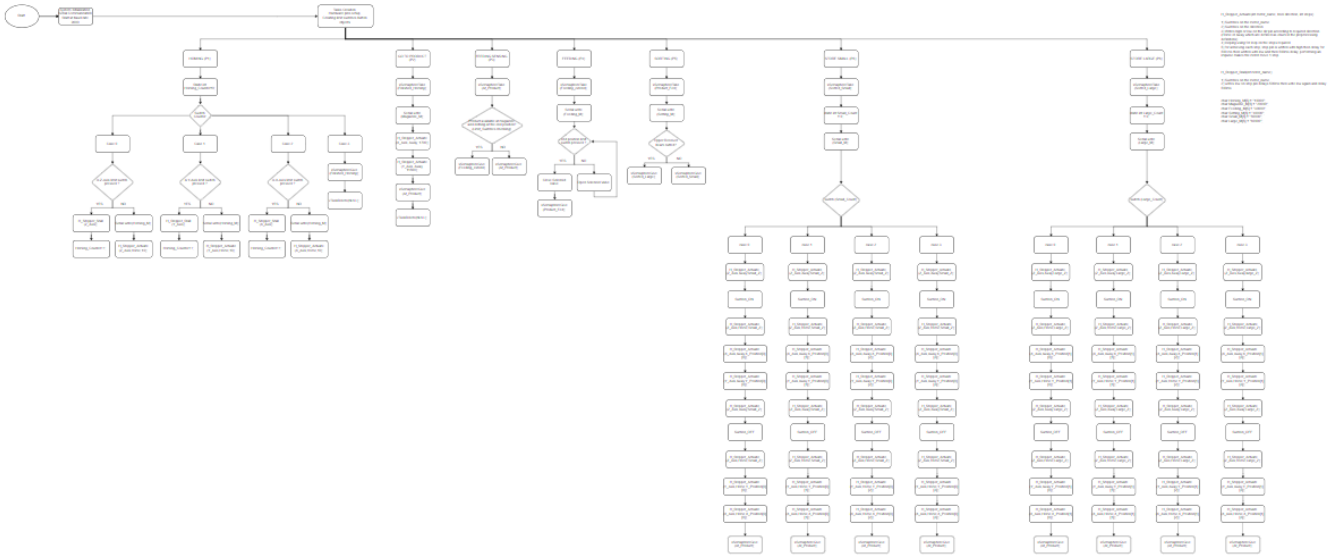


Figure 61: First MCU RTOS FlowChart

3.1.1. System Initialization

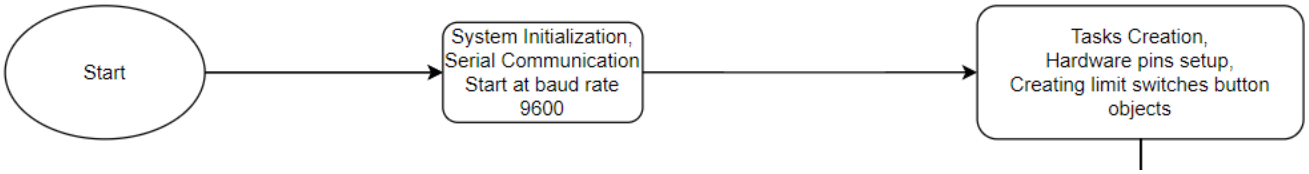


Figure 62: System initialization

3.1.2. Homing Task

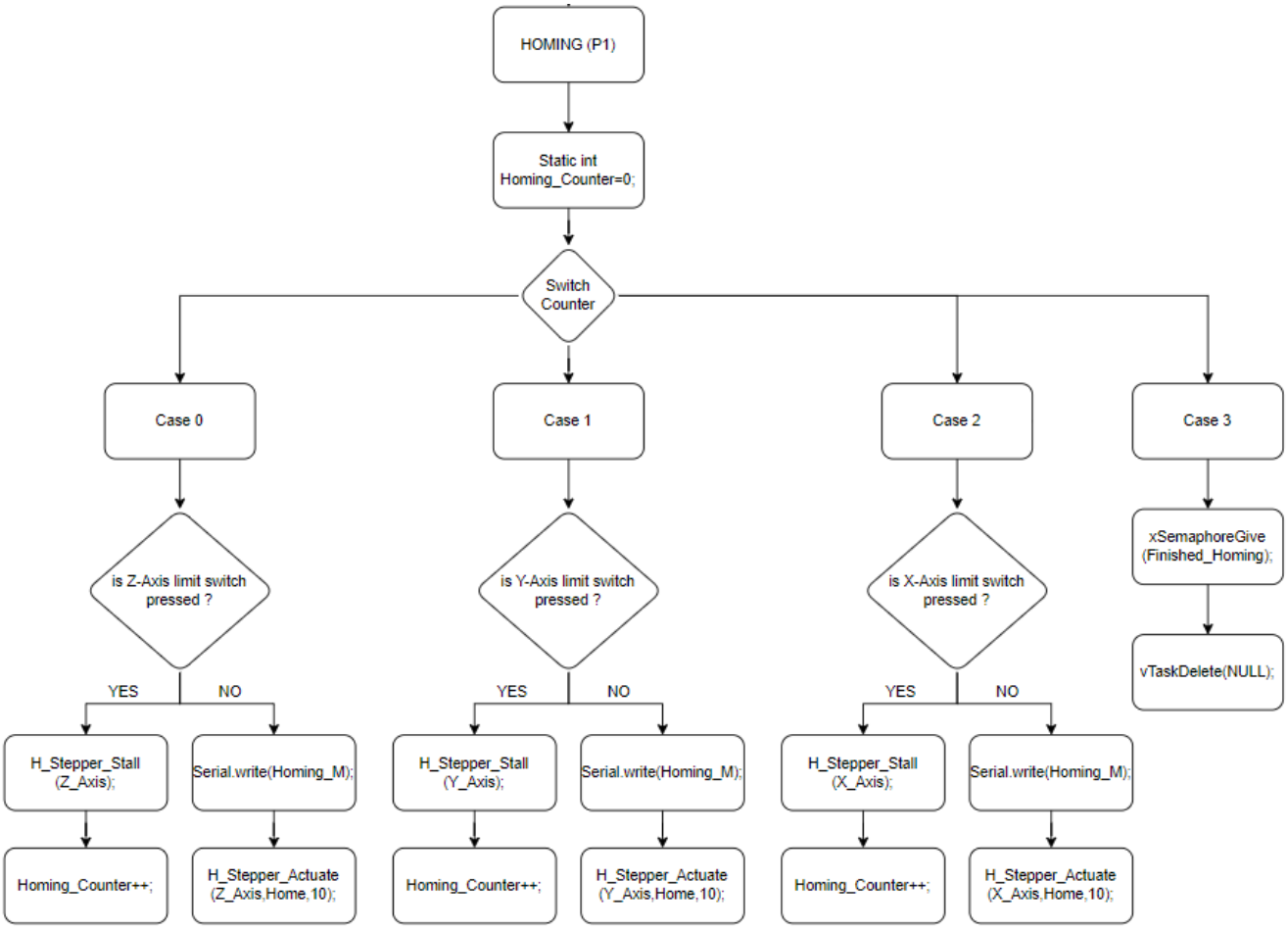


Figure 63: Homing Task

3.1.3. Go To Product Task

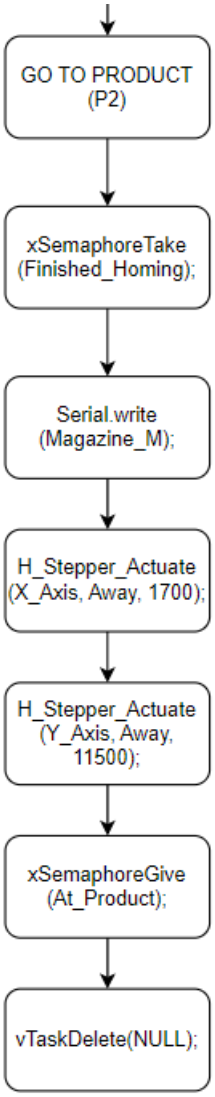


Figure 64: Go To Product Task

3.1.4. Feeding Sensing Task

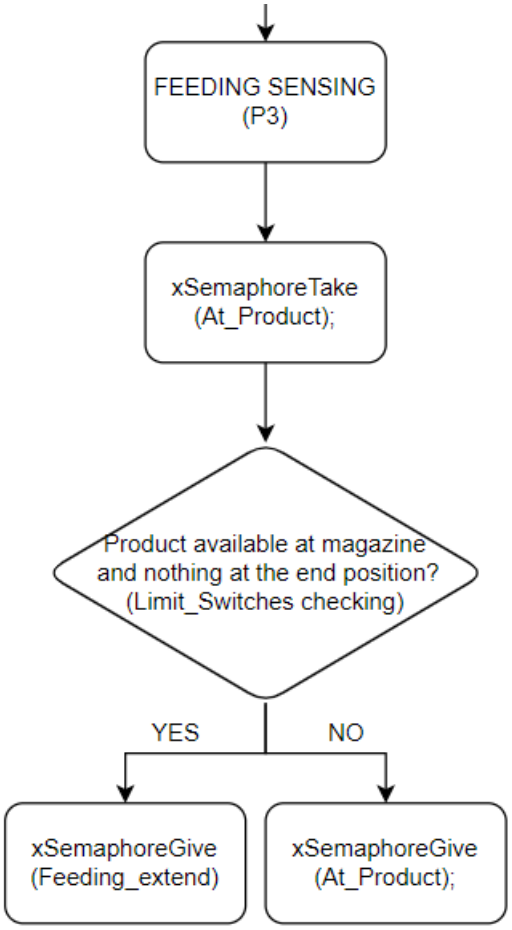


Figure 65: Feeding Sensing Task

3.1.5. Feeding Task

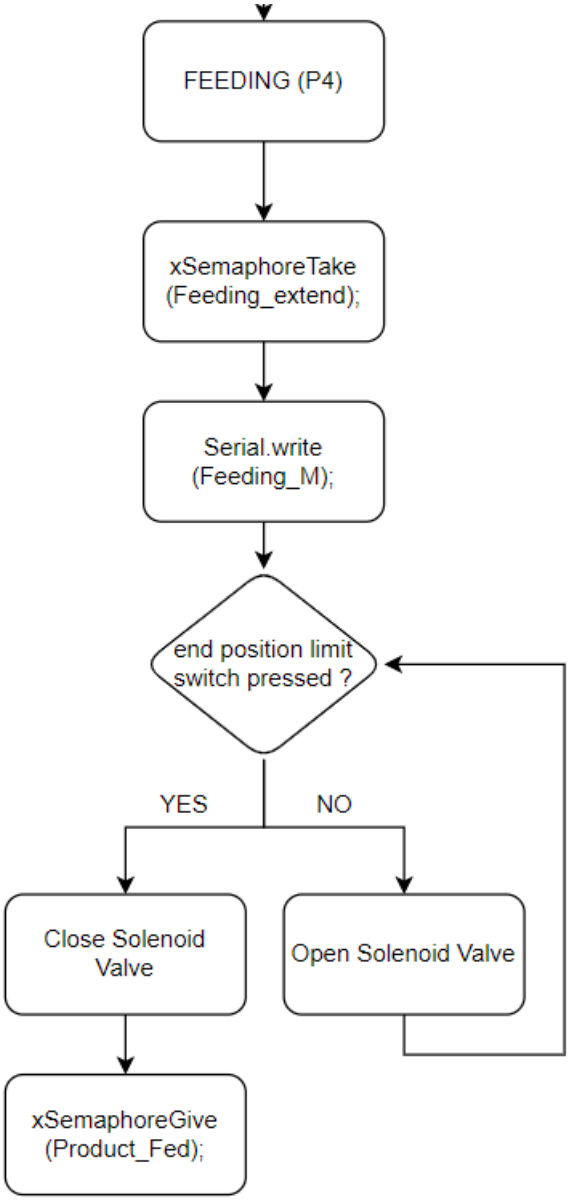


Figure 66: Feeding Task

3.1.6. Sorting Task

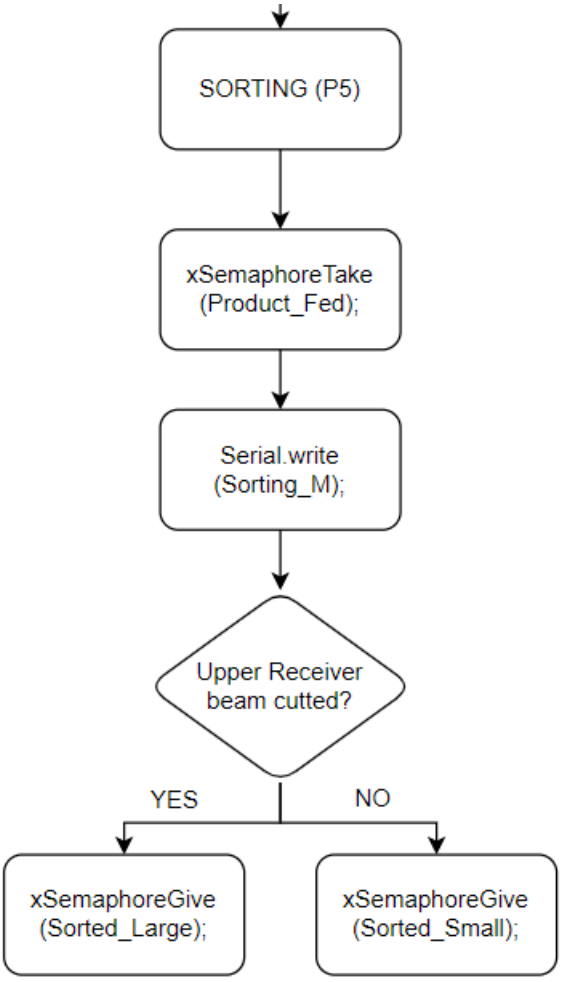


Figure 67: Sorting Task

3.1.7. Large Store Task

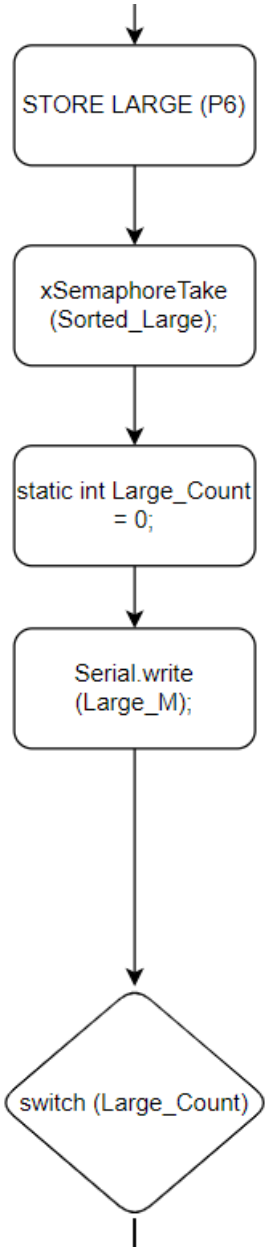


Figure 68: Large Product Storing Task Part1

Figure 69: Large Product Storing Task Part 2

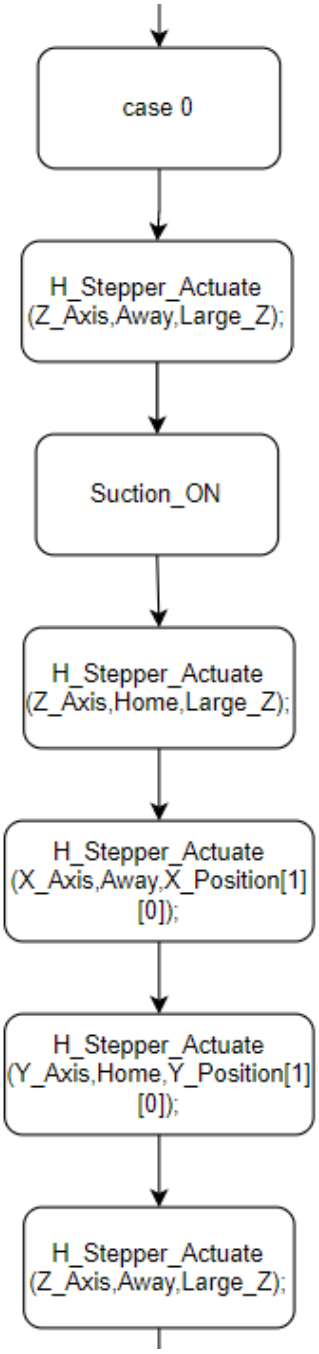


Figure 70: Case Algorithm Part 1

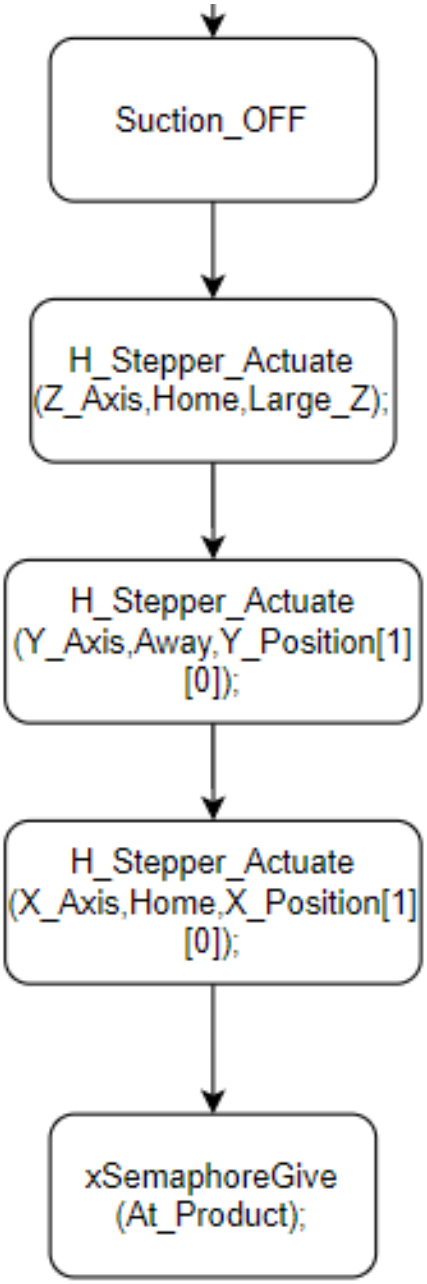


Figure 71: Case Algorithm Part 2

3.1.8. Small Store Task

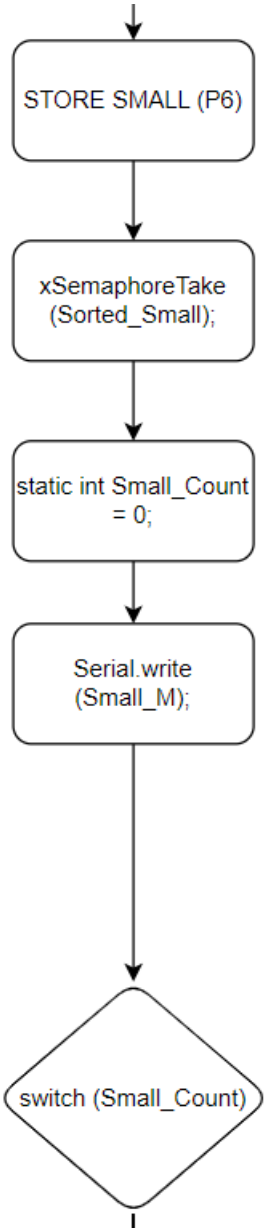


Figure 72: Small Storing Task Part 1

Figure 73: Small Storing Task Part 2

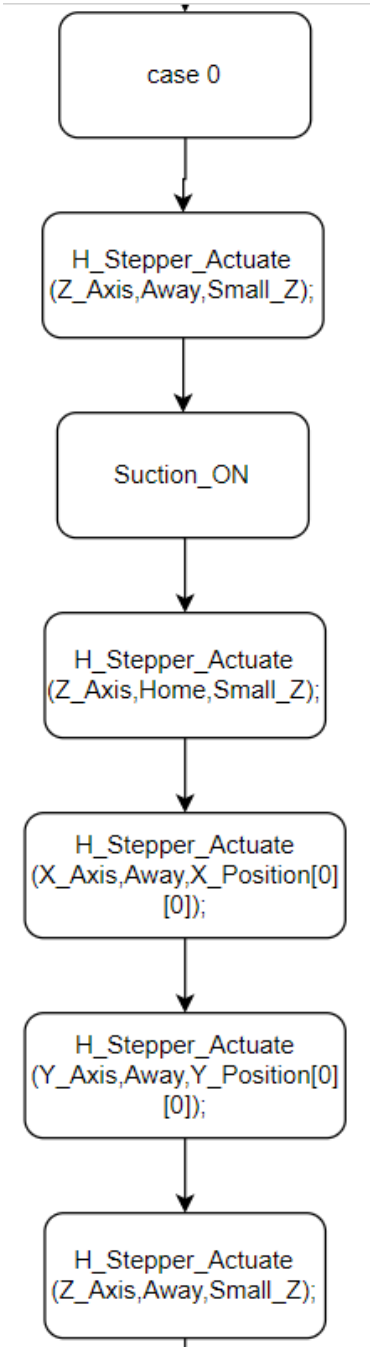


Figure 74: Case Algorithm Part 1

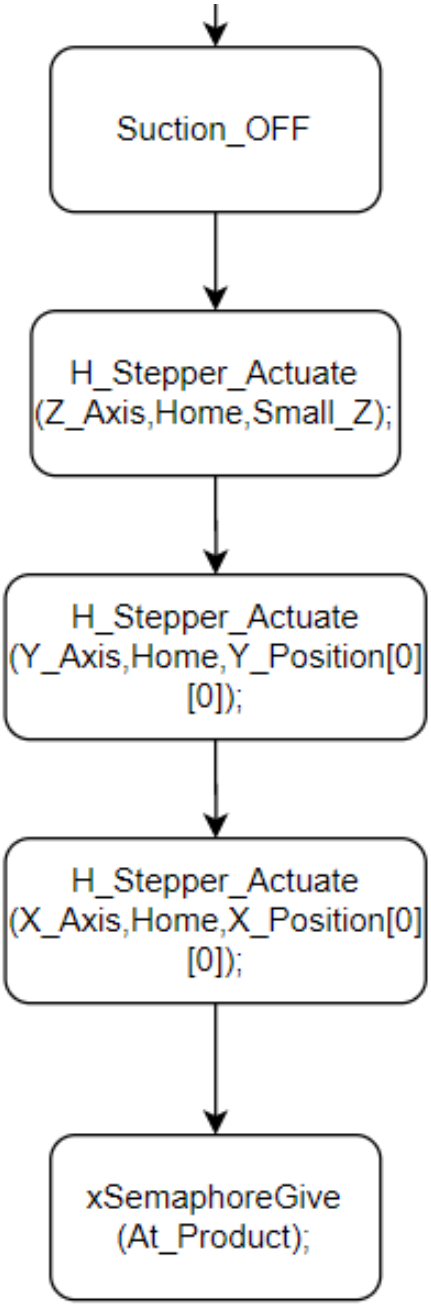


Figure 75: Case Algorithm Part 2

3.2. Second MCU (Micro-Controller Unit)

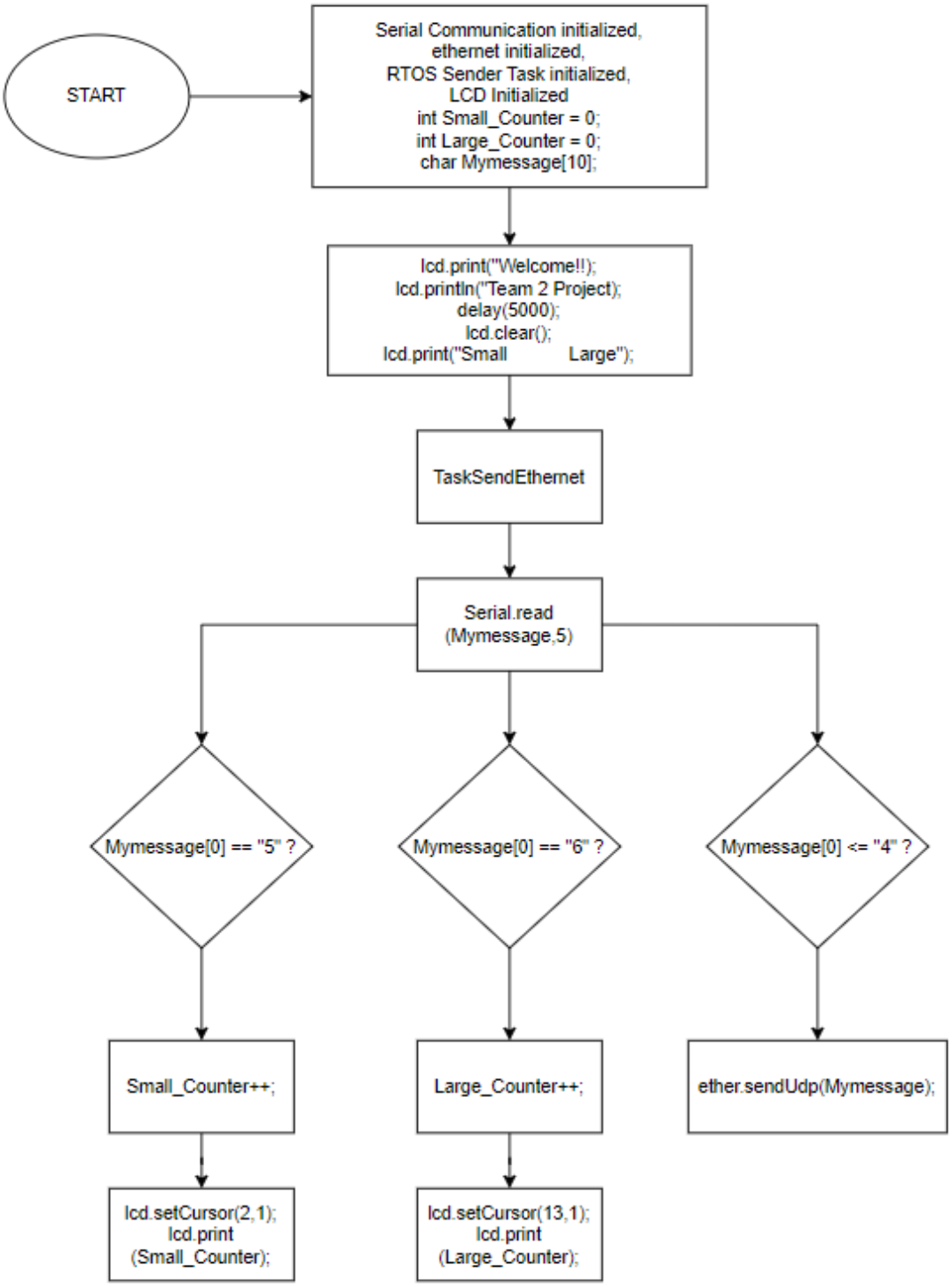


Figure 76: Second MCU RTOS Flowchart

3.3. Third MCU

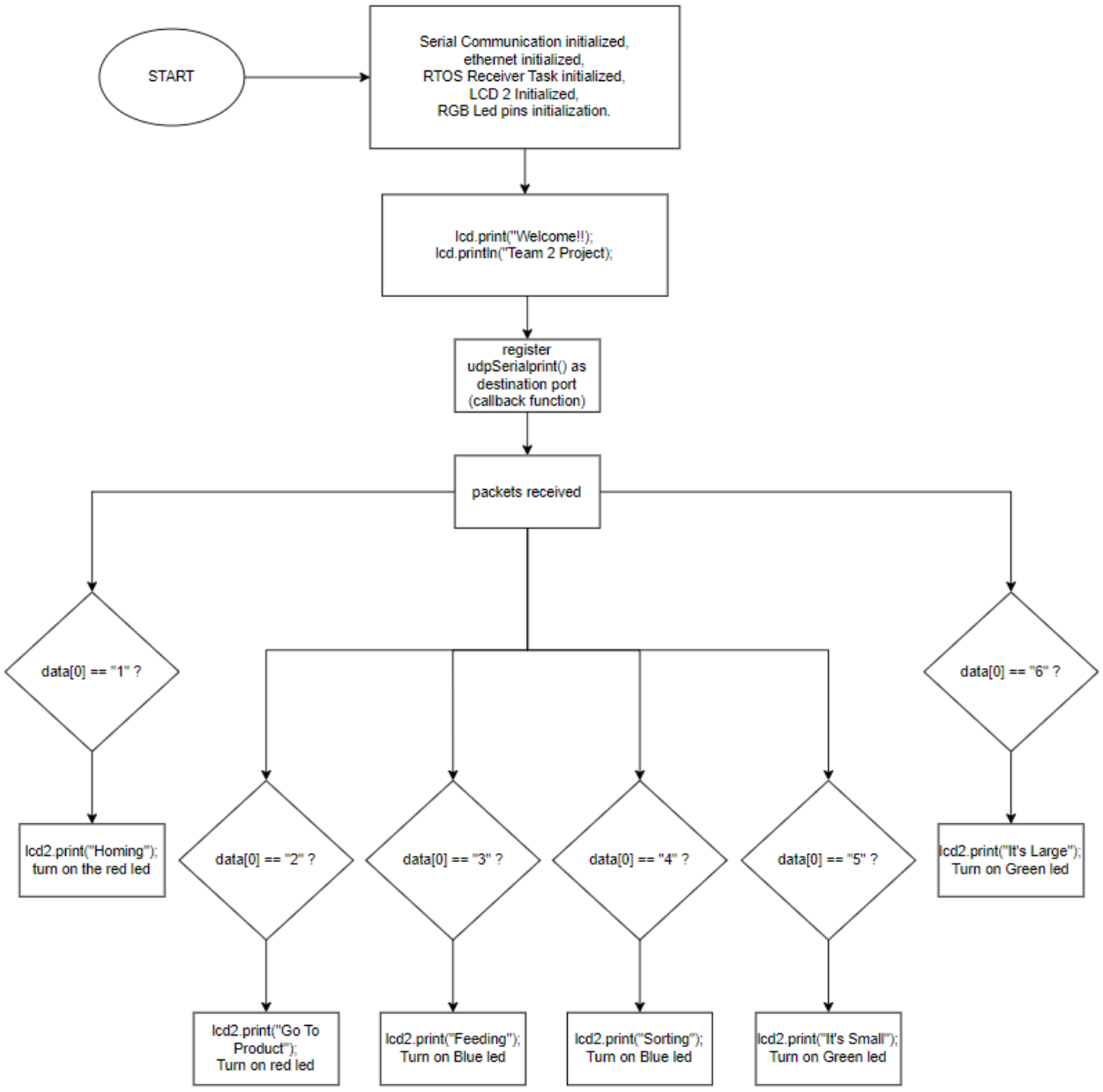


Figure 77: Third MCU RTOS Flowchart

5. RTOS Code

5.1. MCU 1

5.1.1. Code Definition

```
/*********************************************************************************************************************************************
 *
 * FILE-NAME : Arduino_Nano_Code
 *
 * AUTHOR : TEAM 2
 *
 * DESCRIPTION : Arduino Nano code that control all operations and communicate with 2nd MCU using UART Protocols
 *               to print on LCD connected on 2nd MCU the count of products stored in each storage location
 *               and the 2nd MCU act as an bridge for the communication between the 1st and 3rd MCUs to that the 3rd MCU
 *               print on an LCD the currently operating process and control the lights that notifies about the operation being done
 *
 *********************************************************************************************************************************************/

#include "Arduino_FreeRTOS.h"
#include "semphr.h"
#include "ezButton.h"

char Homing_M[5]    = "10000"; //String data
char Magazine_M[5]  = "20000";
char Feeding_M[5]   = "30000";
char Sorting_M[5]   = "40000";
char Small_M[5]     = "50000";
char Large_M[5]     = "60000";

```

Figure 78: Code Definition

5.1.2. Hardware Definition

```
/*********************************************************************************************************************************************
 ****************************************     Pins Declaration      ***************************************************************
 ****************************************    Hardware Definition    ***************************************************************
 *********************************************************************************************************************************************/

/***************************                  Robotic Arm                              *********************************/

ezButton X_Switch(9);              // create ezButton object that attach to pin 9 to act as limit switch for x-axis
ezButton Y_Switch(A0);             // create ezButton object that attach to pin A0 to act as limit switch for y-axis
ezButton Z_Switch(A1);             // create ezButton object that attach to pin A1 to act as limit switch for Z-axis

// declaration of the pins controlling the direction and step of each motor (According to data sheet of CNC NANO SHIELD)
#define X_Dir   2
#define Y_Dir   3
#define Z_Dir   4
#define X_Step  5
#define Y_Step  6
#define Z_Step  7

// defining the storing patch and locations in the production line
const int Products_Types  = 2;         // either Large/Small
const int Products_Count  = 4;         // 4 products of each type

// locations, as stepper motor steps count, defined from top left to bottom right
#define First_Row     29550
#define Second_Row    23300
#define First_Col     8450
#define Second_Col    2400
#define Third_Col     3750
#define Fourth_Col    10000
#define Large_Z       5850
#define Small_Z       6850
int X_Position[Products_Types][Products_Count]= {{First_Row+50,First_Row+50 ,Second_Row ,Second_Row},{First_Row   ,First_Row  ,Second_Row+100 ,Second_Row+100 }};  // saved X-locations
int Y_Position[Products_Types][Products_Count]= {{First_Col   ,Second_Col   ,First_Col  ,Second_Col},{Fourth_Col  ,Third_Col  ,Fourth_Col     ,Third_Col      }};  // saved Y-locations
enum Product_Type {Small,Large};
enum Product_No   {One,Two,Three,Four};

```

Figure 79: Hardware Definition

5.1.3. Feeding and Suction Definition

```
61
62  /***************************                    Feeding & Suction                    **********************************/
63
64  ezButton Feeding_Switch(A3);          // create ezButton object that attach to pin A3 to check that product reached end position of feeding
65  ezButton Product_Switch(A2);          // create ezButton object that attach to pin A2 to check that product is available in magazine
66  #define Suction_DCV 11                // attach the suction DCV pin to pin 11 in shield
67  #define Feeding_DCV 10                // attach the Feeding DCV pin to pin 10 in shield
68  #define Upper_Receiver A6             // attach pin A6 to read value of the upper laser receiver
69
70  enum Feeder_State{OFF,ON};
71  int Large_Count = 0;
72  int Small_Count = 0;
```

Figure 80: Feeding & Suction Defintions

5.1.4. Functions and Semaphores

```
73
74  /*****************************************************************************************************************************
75  ************************************     Functions Declaration      ************************************************************
76  *****************************************************************************************************************************/
77
78  enum Direction{Away, Home};                                 // 2 directions for each axis either towards the Home position or away from it
79  enum Motor{X_Axis, Y_Axis, Z_Axis};                         // 3 overall motors in the line
80  void H_Stepper_Actuate(int Motor, bool Direction, int Steps);     // our customized function for actuating any motor in any direction and with the required steps
81  void H_Stepper_Stall(int Motor);                            // our customized function for stopping any motor
82  void H_Product_Store(bool Product_Type, int Product_Count);       // our customized function for storing any type of product according to count of specific type
83
84  /*****************************************************************************************************************************
85  ************************************       Semaphores        ****************************************************************
86  *****************************************************************************************************************************/
87
88  SemaphoreHandle_t Finished_Homing;
89  SemaphoreHandle_t At_Product;
90  SemaphoreHandle_t Feeding_extend;
91  SemaphoreHandle_t Product_Fed;
92  SemaphoreHandle_t Sorted_Large;
93  SemaphoreHandle_t Sorted_Small;
```

Figure 81: Functions and Semaphores

5.1.5. Setup Function

```
/*****************************************************************************************************************************
*************************************      Setup Function      *****************************************************************
*****************************************************************************************************************************/

// the setup function runs once when you press reset or power the board
void setup() {
  // All Hardware Pins on this microcontroller are outputs except the laser receiver
  pinMode(X_Step        , OUTPUT);
  pinMode(X_Dir         , OUTPUT);
  pinMode(Y_Step        , OUTPUT);
  pinMode(Y_Dir         , OUTPUT);
  pinMode(Z_Step        , OUTPUT);
  pinMode(Z_Dir         , OUTPUT);
  pinMode(Suction_DCV   , OUTPUT);
  pinMode(Feeding_DCV   , OUTPUT);
  pinMode(Upper_Receiver, INPUT );                          // Receiver Declared as input

  // to ensure everything is off on system powering ON
  digitalWrite(Suction_DCV, OFF);
  digitalWrite(Feeding_DCV, OFF);

  Product_Switch.setDebounceTime(50);                       // set debounce time to 50 milliseconds
  Feeding_Switch.setDebounceTime(50);                       // set debounce time to 50 milliseconds

  X_Switch.setDebounceTime(50);                             // set debounce time to 50 milliseconds
  Y_Switch.setDebounceTime(50);                             // set debounce time to 50 milliseconds
  Z_Switch.setDebounceTime(50);                             // set debounce time to 50 milliseconds

  // Serial initialize
  Serial.begin(9600);

  // Semaphores creation
  Finished_Homing = xSemaphoreCreateBinary();
  At_Product      = xSemaphoreCreateBinary();
  Feeding_extend  = xSemaphoreCreateBinary();
  Product_Fed     = xSemaphoreCreateBinary();
  Sorted_Large    = xSemaphoreCreateBinary();
  Sorted_Small    = xSemaphoreCreateBinary();

  xTaskCreate(Homing          , "Task1", 100, NULL, 1, NULL);
  xTaskCreate(GoToProduct     , "Task2", 100, NULL, 2, NULL);
  xTaskCreate(Feeding_Sensing , "Task3", 100, NULL, 3, NULL);
  xTaskCreate(Feeding         , "Task4", 100, NULL, 4, NULL);
  xTaskCreate(Sorting         , "Task5", 100, NULL, 5, NULL);
  xTaskCreate(Small_Storing   , "Task6", 100, NULL, 6, NULL);
  xTaskCreate(Large_Storing   , "Task7", 100, NULL, 6, NULL);
}
```

Figure 82: Setup function

5.1.6. Homing Task

```
151
152  /*****************************************************************************************************************************
153  *********************************      Customized RTOS Functions      ********************************************************
154  *****************************************************************************************************************************/
155
156  // Function that aims to Home the Robotic Arm at Powering up or Reset
157  void Homing (void* ptr)
158  {
159    while(1)
160    {
161
162      static int Homing_Counter = 0;                        // counter to check that an axis has homed
163      X_Switch.loop();                                      // MUST call the loop() function first
164      Y_Switch.loop();                                      // MUST call the loop() function first
165      Z_Switch.loop();                                      // MUST call the loop() function first
166      switch(Homing_Counter)
167      {
168        case 0:
169        // Z-Axis Homing Loop (Done First to avoid Hitting objects while moving)
170        if(Z_Switch.isPressed() || !Z_Switch.getState())
171        {
172          Homing_Counter ++;
173          H_Stepper_Stall(Z_Axis);
174        }
175        else
176        {
177          H_Stepper_Actuate(Z_Axis, Home, 10);
178        }
179        break;
180
181        case 1:
182        // Y-Axis Homing Loop (Second thing done to avoid hitting magazine while moving)
183        if(Y_Switch.isPressed() || !Y_Switch.getState())
184        {
185          Homing_Counter ++;
186          H_Stepper_Stall(Y_Axis);
187        }
188        else
189        {
190          H_Stepper_Actuate(Y_Axis, Home, 10);
191        }
192        break;
```

Figure 83: Homing Task

```
      case 2:
      // X-Axis Homing Loop
      if(X_Switch.isPressed() || !X_Switch.getState())
      {
        Homing_Counter ++;
        H_Stepper_Stall(X_Axis);
      }
      else
      {
        H_Stepper_Actuate(X_Axis, Home, 2);
      }
      break;

      case 3:
      xSemaphoreGive(Finished_Homing);
      vTaskDelete(NULL);
      break;
    }
  }
}
```

Figure 84: Homing Task Part 2

5.1.7. Go to Product Task

```
/*****************************************************************************************************************************************************
******************************************************************************************************************************************************/

// Function that aims to stand above the product location in the magazine after being Homed at the power up or reset
void GoToProduct (void* ptr)
{
  while(1)
  {
    xSemaphoreTake(Finished_Homing,portMAX_DELAY);
    Serial.write(Magazine_M,5); //Write the serial data
    delay(1000);

    H_Stepper_Actuate(X_Axis, Away, 1700);
    H_Stepper_Actuate(Y_Axis, Away, 11500);

    xSemaphoreGive(At_Product);
    vTaskDelete(NULL);
  }
}
```

Figure 85: Go to Product Task

5.1.8. Feeding Sensing Task

```
/*****************************************************************************************************************************************************
******************************************************************************************************************************************************/

// Function that checks if there's a product in the magazine or not and check if feeding is appropriate at the moment
void Feeding_Sensing (void* ptr)
{
  while(1)
  {
    xSemaphoreTake(At_Product,portMAX_DELAY);
    Product_Switch.loop();                              // MUST call the loop() function first
    Feeding_Switch.loop();

    int endposition = Feeding_Switch.getState();
    int available   = Product_Switch.getState();

    if((available == LOW) && (endposition == HIGH))     // product available at the magazine and nothing at the end position
    {
      xSemaphoreGive(Feeding_extend);
    }
  }
}
```

Figure 86: Feeding Sensing Task

5.1.9. Feeding Task

```
/*****************************************************************************************************************************************************
******************************************************************************************************************************************************/

// Function that feed a product into the sorting terminal
void Feeding (void* ptr)
{
  while(1)
  {

    xSemaphoreTake(Feeding_extend, portMAX_DELAY);
    Serial.write(Feeding_M,5); //Write the serial data

    digitalWrite(Feeding_DCV,ON);
    delay(1000);
    digitalWrite(Feeding_DCV,OFF);
    xSemaphoreGive(Product_Fed);

  }
}
```

Figure 87: Feeding Task

5.1.10. Sorting Task

```
/*****************************************************************************************************************************************************
******************************************************************************************************************************************************/

// Function that Sort the product available at sorting channel according to type (Large/Small)
void Sort (void* ptr)
{
  while(1)
  {
    xSemaphoreTake(Product_Fed,portMAX_DELAY);
    Serial.write(Sorting_M,5); //Write the serial data
    delay(1000);
    if(analogRead(Upper_Receiver)==0)
    {
      Small_Count++;
      xSemaphoreGive(Sorted_Small);
    }
    else
    {
      Large_Count++;
      xSemaphoreGive(Sorted_Large);
    }
  }
}
```

Figure 88: Sorting Task

5.1.11. Large Storing Task

```
/*****************************************************************************************************************************************************
******************************************************************************************************************************************************/

// Function that store the large products according to the number of the product according to the past processes
void Large_Storing (void* ptr)
{
  while(1)
  {
    xSemaphoreTake(Sorted_Large,portMAX_DELAY);
    Serial.write(Large_M,5); //Write the serial data
    delay(1000);
    H_Product_Store(Large, Large_Count);
    xSemaphoreGive(At_Product);
  }
}
```

Figure 89: Large Storing Task

5.1.12. Small Storing Task

```
/*****************************************************************************************************************************************************
******************************************************************************************************************************************************/

// Function that store the Small products according to the number of the product according to the past processes
void Small_Storing (void* ptr)
{
  while(1)
  {
    xSemaphoreTake(Sorted_Large,portMAX_DELAY);
    Serial.write(Small_M,5); //Write the serial data
    delay(1000);
    H_Product_Store(Small, Small_Count);
    xSemaphoreGive(At_Product);
  }
}
```

Figure 90: Small Storing Task

5.1.13. Special Functions

```
/*****************************************************************************************************************
************************************      Customized  Functions      ************************************************
*****************************************************************************************************************/

// our customized function that actuate any motor in the project in both direction with the favoured steps count
void H_Stepper_Actuate(int Motor, bool Direction, int Steps)
{
  switch(Motor)
  {
    case X_Axis:
    switch(Direction)
    {
      case Away:
      digitalWrite(X_Dir, LOW);
      for (int x = 0;x<Steps; x++)
      {
        digitalWrite(X_Step,HIGH);
        delayMicroseconds(500);
        digitalWrite(X_Step,LOW);
        delayMicroseconds(500);
      }
      break;
      case Home:
      digitalWrite(X_Dir, HIGH);
      for (int x = 0;x<Steps; x++)
      {
        digitalWrite(X_Step,HIGH);
        delayMicroseconds(500);
        digitalWrite(X_Step,LOW);
        delayMicroseconds(500);
      }
      break;
    }
    break;
```

Figure 91: H_Stepper_Actuate Part 1

```
367     case Y_Axis:
368     switch(Direction)
369     {
370       case Away:
371       digitalWrite(Y_Dir, LOW);
372       for (int y = 0;y<Steps; y++)
373       {
374         digitalWrite(Y_Step,HIGH);
375         delayMicroseconds(500);
376         digitalWrite(Y_Step,LOW);
377         delayMicroseconds(500);
378       }
379       break;
380       case Home:
381       digitalWrite(Y_Dir, HIGH);
382       for (int y = 0;y<Steps; y++)
383       {
384         digitalWrite(Y_Step,HIGH);
385         delayMicroseconds(500);
386         digitalWrite(Y_Step,LOW);
387         delayMicroseconds(500);
388       }
389       break;
390     }
391     break;
392     case Z_Axis:
393     switch(Direction)
394     {
395       case Away:
396       digitalWrite(Z_Dir, LOW);
397       for (int z = 0;z<Steps; z++)
398       {
399         digitalWrite(Z_Step,HIGH);
400         delayMicroseconds(500);
401         digitalWrite(Z_Step,LOW);
402         delayMicroseconds(500);
403       }
404       break;
405       case Home:
406       digitalWrite(Z_Dir, HIGH);
407       for (int z = 0;z<Steps; z++)
408       {
409         digitalWrite(Z_Step,HIGH);
410         delayMicroseconds(500);
411         digitalWrite(Z_Step,LOW);
412         delayMicroseconds(500);
413       }
414       break;
415     }
```

Figure 92: H_Stepper_Actuate Part 2

```
/*****************************************************************************************************************************
*****************************************************************************************************************************/

// our customized function that stops any motor in the project
void H_Stepper_Stall(int Motor)
{
  switch(Motor)
  {
    case X_Axis:
    digitalWrite(X_Step,LOW);
    delayMicroseconds(500);
    digitalWrite(X_Step,LOW);
    delayMicroseconds(500);
    break;
    case Y_Axis:
    digitalWrite(Y_Step,LOW);
    delayMicroseconds(500);
    digitalWrite(Y_Step,LOW);
    delayMicroseconds(500);
    break;
    case Z_Axis:
    digitalWrite(Z_Step,LOW);
    delayMicroseconds(500);
    digitalWrite(Z_Step,LOW);
    delayMicroseconds(500);
    break;
  }
}
```

Figure 93: H_Stepper_Stall

```
/*****************************************************************************************************************
*****************************************************************************************************************/

// our customized function for storing any type of product according to count of specific type
// the commenting of the first case applies to all cases in the function but varies in the positions to be gone or stroke of Z-Axis for suction mechanism

void H_Product_Store(bool Product_Type, int Product_Count)
{
  switch(Product_Type)
  {
    case Large:
      switch(Product_Count)
      {
        case 1:
        H_Stepper_Actuate(Z_Axis,Away,Large_Z);                 // Goes down
        delay(500);                                             // waits half a second
        digitalWrite(Suction_DCV,ON);                           // turn on the suction
        delay(500);                                             // waits half a second
        H_Stepper_Actuate(Z_Axis,Home,Large_Z);                 // goes up

        H_Stepper_Actuate(X_Axis,Away,X_Position[Large][One]);  // goes to the row of the product to be stored
        H_Stepper_Actuate(Y_Axis,Home,Y_Position[Large][One]);  // goes to the column of the product to be stored

        H_Stepper_Actuate(Z_Axis,Away,Large_Z);                 // goes down
        delay(500);                                             // waits half a second
        digitalWrite(Suction_DCV,OFF);                          // turn off the suction
        delay(500);                                             // waits half a second
        H_Stepper_Actuate(Z_Axis,Home,Large_Z);                 // goes up

        H_Stepper_Actuate(Y_Axis,Away,Y_Position[Large][One]);  // return in y axis first
        H_Stepper_Actuate(X_Axis,Home,X_Position[Large][One]);  // then returns in the X-Axis to be standing above the product again
        break;
        case 2:
        H_Stepper_Actuate(Z_Axis,Away,Large_Z);
        delay(500);
        digitalWrite(Suction_DCV,ON);
        delay(500);
        H_Stepper_Actuate(Z_Axis,Home,Large_Z);

        H_Stepper_Actuate(X_Axis,Away,X_Position[Large][Two]);
        H_Stepper_Actuate(Y_Axis,Home,Y_Position[Large][Two]);

        H_Stepper_Actuate(Z_Axis,Away,Large_Z);
        delay(500);
        digitalWrite(Suction_DCV,OFF);
        delay(500);
        H_Stepper_Actuate(Z_Axis,Home,Large_Z);

        H_Stepper_Actuate(Y_Axis,Away,Y_Position[Large][Two]);
        H_Stepper_Actuate(X_Axis,Home,X_Position[Large][Two]);
        break;
```

Figure 94: H_Product_Store Part 1

```
500         case 3:
501         H_Stepper_Actuate(Z_Axis,Away,Large_Z);
502         delay(500);
503         digitalWrite(Suction_DCV,ON);
504         delay(500);
505         H_Stepper_Actuate(Z_Axis,Home,Large_Z);
506
507         H_Stepper_Actuate(X_Axis,Away,X_Position[Large][Three]);
508         H_Stepper_Actuate(Y_Axis,Home,Y_Position[Large][Three]);
509
510         H_Stepper_Actuate(Z_Axis,Away,Large_Z);
511         delay(500);
512         digitalWrite(Suction_DCV,OFF);
513         delay(500);
514         H_Stepper_Actuate(Z_Axis,Home,Large_Z);
515
516         H_Stepper_Actuate(Y_Axis,Away,Y_Position[Large][Three]);
517         H_Stepper_Actuate(X_Axis,Home,X_Position[Large][Three]);
518         break;
519         case 4:
520         H_Stepper_Actuate(Z_Axis,Away,Large_Z);
521         delay(500);
522         digitalWrite(Suction_DCV,ON);
523         delay(500);
524         H_Stepper_Actuate(Z_Axis,Home,Large_Z);
525
526         H_Stepper_Actuate(X_Axis,Away,X_Position[Large][Four]);
527         H_Stepper_Actuate(Y_Axis,Home,Y_Position[Large][Four]);
528
529         H_Stepper_Actuate(Z_Axis,Away,Large_Z);
530         delay(500);
531         digitalWrite(Suction_DCV,OFF);
532         delay(500);
533         H_Stepper_Actuate(Z_Axis,Home,Large_Z);
534
535         H_Stepper_Actuate(Y_Axis,Away,Y_Position[Large][Four]);
536         H_Stepper_Actuate(X_Axis,Home,X_Position[Large][Four]);
537         break;
538       }
```

Figure 95: H_Product_Store Part 2

```
540     case Small:
541       switch(Product_Count)
542       {
543         case 1:
544         H_Stepper_Actuate(Z_Axis,Away,Small_Z);
545         delay(500);
546         digitalWrite(Suction_DCV,ON);
547         delay(500);
548         H_Stepper_Actuate(Z_Axis,Home,Small_Z);
549
550         H_Stepper_Actuate(X_Axis,Away,X_Position[Small][One]);
551         H_Stepper_Actuate(Y_Axis,Away,Y_Position[Small][One]);
552
553         H_Stepper_Actuate(Z_Axis,Away,Small_Z);
554         delay(500);
555         digitalWrite(Suction_DCV,OFF);
556         delay(500);
557         H_Stepper_Actuate(Z_Axis,Home,Small_Z);
558
559         H_Stepper_Actuate(Y_Axis,Home,Y_Position[Small][One]);
560         H_Stepper_Actuate(X_Axis,Home,X_Position[Small][One]);
561         break;
562         case 2:
563         H_Stepper_Actuate(Z_Axis,Away,Small_Z);
564         delay(500);
565         digitalWrite(Suction_DCV,ON);
566         delay(500);
567         H_Stepper_Actuate(Z_Axis,Home,Small_Z);
568
569         H_Stepper_Actuate(X_Axis,Away,X_Position[Small][Two]);
570         H_Stepper_Actuate(Y_Axis,Away,Y_Position[Small][Two]);
571
572         H_Stepper_Actuate(Z_Axis,Away,Small_Z);
573         delay(500);
574         digitalWrite(Suction_DCV,OFF);
575         delay(500);
576         H_Stepper_Actuate(Z_Axis,Home,Small_Z);
577
578         H_Stepper_Actuate(Y_Axis,Home,Y_Position[Small][Two]);
579         H_Stepper_Actuate(X_Axis,Home,X_Position[Small][Two]);
580         break;
```

Figure 96: H_Product_Store Part 3

```
581             case 3:
582             H_Stepper_Actuate(Z_Axis,Away,Small_Z);
583             delay(500);
584             digitalWrite(Suction_DCV,ON);
585             delay(500);
586             H_Stepper_Actuate(Z_Axis,Home,Small_Z);
587
588             H_Stepper_Actuate(X_Axis,Away,X_Position[Small][Three]);
589             H_Stepper_Actuate(Y_Axis,Away,Y_Position[Small][Three]);
590
591             H_Stepper_Actuate(Z_Axis,Away,Small_Z);
592             delay(500);
593             digitalWrite(Suction_DCV,OFF);
594             delay(500);
595             H_Stepper_Actuate(Z_Axis,Home,Small_Z);
596
597             H_Stepper_Actuate(Y_Axis,Home,Y_Position[Small][Three]);
598             H_Stepper_Actuate(X_Axis,Home,X_Position[Small][Three]);
599             break;
600             case 4:
601             H_Stepper_Actuate(Z_Axis,Away,Small_Z);
602             delay(500);
603             digitalWrite(Suction_DCV,ON);
604             delay(500);
605             H_Stepper_Actuate(Z_Axis,Home,Small_Z);
606
607             H_Stepper_Actuate(X_Axis,Away,X_Position[Small][Four]);
608             H_Stepper_Actuate(Y_Axis,Away,Y_Position[Small][Four]);
609
610             H_Stepper_Actuate(Z_Axis,Away,Small_Z);
611             delay(500);
612             digitalWrite(Suction_DCV,OFF);
613             delay(500);
614             H_Stepper_Actuate(Z_Axis,Home,Small_Z);
615
616             H_Stepper_Actuate(Y_Axis,Home,Y_Position[Small][Four]);
617             H_Stepper_Actuate(X_Axis,Home,X_Position[Small][Four]);
618             break;
619           }
620         break;
621       }
622     }
```

Figure 97: H_Product_Store Part 4

5.2. Second MCU

```
/*****************************************************************************************************************************
*
* FILE-NAME : Arduino_UNO_Transition_Code
*
* AUTHOR : TEAM 2
*
* DESCRIPTION : Arduino Uno Code that targets as a bridge for the communication between the 1st MCU and the 3rd one
*               it communicates with 1st MCU using UART protocol and with the 3rd MCU using SPI-Ethernet communication
*               In addition, according to what received from the 1st MCU this MCU displays count of products stored on an LCD
*
*****************************************************************************************************************************/

#include <EtherCard.h>
#include <Arduino_FreeRTOS.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>


/*****************************************************************************************************************************
***************************************     Pins Declaration      ***************************************************************
***************************************    Hardware Definition    ***************************************************************
*****************************************************************************************************************************/

int small = 0;                                          // Small Products Counter
int large =0;                                           // Large Products Counter
/* Addresses */
// mac address
static byte mymac[] = { 0x1A,0x2B,0x3C,0x4D,0x5E,0x6F };
// ethernet interface ip address
static byte myip[] = { 192, 168, 1, 2 };
// gateway ip address
static byte gwip[] = { 192, 168, 1, 1 };
// subnet mask
static byte mask[] = { 255, 255, 255, 0 };
// destination ip-address
static byte dstIp[] = { 192, 168, 1, 3 };
// ports
const int dstPort PROGMEM = 1234;
const int srcPort PROGMEM = 4321;

// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

char Mymessage[10]; //Initialized variable to store recieved data

/* Buffer and timer */
byte Ethernet::buffer[700];
static uint32_t timer;  // dummy incrementor that'll later be used for delays

```

Figure 98: Transition Arduino Code Part 1

```
/*****************************************************************************************************************************
***********************************     Functions Declaration      ***********************************************************
*****************************************************************************************************************************/

/* Ethernet peripheral initialization function */
void initializeEthernet()
{
  // Begin Ethernet communication
  if (ether.begin(sizeof Ethernet::buffer, mymac, SS) == 0)
  {
    Serial.println("Failed to access Ethernet controller");
    return;
  }

  // Setup static IP address
  ether.staticSetup(myip, gwip, 0, mask);

  // Log configuration
  Serial.println(F("\n[Sender]"));
  ether.printIp("IP:  ", ether.myip);
  ether.printIp("GW:  ", ether.gwip);
}

/* FreeRTOS Ethernet Sending Task */
void TaskSendEthernet( void* pvParameters )
{
  while(1)
  {
    Serial.readBytes(Mymessage,5); //Read the serial data and store in var
    ether.sendUdp(Mymessage, sizeof(Mymessage), srcPort, dstIp, dstPort);
    if(Mymessage[0] == "5")
    {
      small++;
      lcd.setCursor(2,1);
      lcd.print(small);
    }
    else if (Mymessage[0] == "6")
    {
      large++;
      lcd.setCursor(13,1);
      lcd.print(large);
    }
  }
}
```

Figure 99: Transition Arduino Code Part 2

```
95
96  /*****************************************************************************************************************************
97  ************************************      Setup Function      ****************************************************************
98  *****************************************************************************************************************************/
99
100 void setup () {
101   // Initialize Serial
102   Serial.begin(9600);
103
104   // Initialize Ethernet
105   initializeEthernet();
106
107   lcd.begin();
108   lcd.backlight();
109
110   lcd.setCursor(0, 0);
111   lcd.print("Welcome!!");
112   lcd.setCursor(0, 1);
113   lcd.print("Team 2 Project");
114   delay(2000);
115   lcd.clear();
116   lcd.print("Small     Large");
117
118   // initialize RTOS task
119   xTaskCreate(
120     TaskSendEthernet
121     ,  "SendEthernet"  // A name just for humans
122     ,  128  // This stack size can be checked & adjusted by reading the Stack Highwater
123     ,  NULL //Parameters for the task
124     ,  2  // Priority, with 3 (configMAX_PRIORITIES - 1) being the highest, and 0 being the lowest.
125     ,  NULL ); //Task Handle
126
127 }
128
129 /*****************************************************************************************************************************
130 ************************************      LOOP Function      ****************************************************************
131 *****************************************************************************************************************************/
132
133 void loop ()
134 {
135   // empty loop
136 }
137
```

Figure 100: Arduino Transition Code Part 3

5.3. Third MCU

```
/*****************************************************************************************************************************
*
* FILE-NAME : Arduino_UNO_Terminal_Code
*
* AUTHOR : TEAM 2
*
* DESCRIPTION : Arduino Uno Code that targets to display the operating process on an LCD according to received data from MCU2 using
*               SPI-Ethernet communication that was previously received from MCU1 to MCU2 using UART
*               in addition it lights different colors from a RGB LED to notify about the momentarily process
*
*****************************************************************************************************************************/

#include <EtherCard.h>
#include <IPAddress.h>
#include <LiquidCrystal.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>


/*****************************************************************************************************************************
***************************************     Pins Declaration      ***************************************************************
**************************************    Hardware Definition    ***************************************************************
*****************************************************************************************************************************/

// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd2(0x27, 16, 2);

#define Red 7
#define Green 6
#define Blue 5


/* Addresses */
// mac address
static byte mymac[] = { 0x70,0x69,0x69,0x2D,0x30,0x31 };
// ethernet interface ip address
static byte myip[] = { 192, 168, 1, 3 };
// gateway ip address
static byte gwip[] = { 192, 168, 1, 1 };
// subnet mask
static byte mask[] = { 255, 255, 255, 0 };
// ports
const int dstPort PROGMEM = 1234;

/* Buffer */
byte Ethernet::buffer[700];

```

Figure 101: Arduino terminal COde Part 1

```
/*****************************************************************************************************************
***********************************     Functions Declaration      ***********************************************
*****************************************************************************************************************/

void initializeEthernet()
{
  // Begin Ethernet communication
  if (ether.begin(sizeof Ethernet::buffer, mymac, SS) == 0)
  {
    Serial.println("Failed to access Ethernet controller");
    return;
  }

  // Setup static IP address
  ether.staticSetup(myip, gwip, 0, mask);

  // Log configuration
  Serial.println(F("\n[Receiver]"));
  ether.printIp("IP:  ", ether.myip);
  ether.printIp("GW:  ", ether.gwip);
}

/* callback that prints received packets to the serial port */
void udpSerialPrint(uint16_t dest_port, uint8_t src_ip[IP_LEN], uint16_t src_port, const char *data, uint16_t len){
  IPAddress src(src_ip[0],src_ip[1],src_ip[2],src_ip[3]);

  Serial.print("dest_port: ");
  Serial.println(dest_port);
  Serial.print("src_port: ");
  Serial.println(src_port);

  Serial.print("src_port: ");
  ether.printIp(src_ip);
  Serial.println("\ndata: ");
  Serial.println(data);
  if( data[0] == '1'){
    lcd2.setCursor(0,0);
    lcd2.print("HOMING          ");
    digitalWrite(Red,HIGH);
    digitalWrite(Green,LOW);
    digitalWrite(Blue,LOW);
  }
  else if( data[0] == '2'){
    lcd2.setCursor(0,0);
    lcd2.print("GO TO PRODUCT   ");
    digitalWrite(Red,HIGH);
    digitalWrite(Green,LOW);
    digitalWrite(Blue,LOW);
  }
```

Figure 102: Arduino Terminal Code Part 2

```
  }
  else if( data[0] == '3'){
    lcd2.setCursor(0,0);
    lcd2.print("FEEDING         ");
    digitalWrite(Red,LOW);
    digitalWrite(Green,LOW);
    digitalWrite(Blue,HIGH);
  }
  else if( data[0] == '4'){
    lcd2.setCursor(0,0);
    lcd2.print("SORTING         ");
    digitalWrite(Red,HIGH);
    digitalWrite(Green,LOW);
    digitalWrite(Blue,HIGH);
  }
   else if( data[0] == '5'){
    lcd2.setCursor(0,0);
    lcd2.print("STORING         ");
    lcd2.setCursor(0,1);
    lcd2.print("SMALL PRODUCT   ");
    digitalWrite(Red,LOW);
    digitalWrite(Green,HIGH);
    digitalWrite(Blue,LOW);
  }
  else if( data[0] == '6'){
    lcd2.setCursor(0,0);
    lcd2.print("STORING         ");
    lcd2.setCursor(0,1);
    lcd2.print("LARGE PRODUCT   ");
    digitalWrite(Red,LOW);
    digitalWrite(Green,HIGH);
    digitalWrite(Blue,LOW);
  }
}
```

Figure 103: Arduino terminal Code Part 3

```
/*****************************************************************************************************************************
*************************************      Setup Function      ***************************************************************
*****************************************************************************************************************************/

void setup(){

  // Initialize Serial
  Serial.begin(9600);

  //lcd1.begin(16,2);
  //lcd1.clear();

  lcd2.begin();

  lcd2.backlight();

  pinMode(Red,OUTPUT);
  pinMode(Green,OUTPUT);
  pinMode(Blue,OUTPUT);


  // Initialize Ethernet
  initializeEthernet();

  //register udpSerialPrint() to destination port (callback function)
  ether.udpServerListenOnPort(&udpSerialPrint, dstPort);
  lcd2.setCursor(0, 0);
  lcd2.print("Welcome!!");
  lcd2.setCursor(0, 1);
  lcd2.print("Team 2 Project");
  delay(2000);
  lcd2.clear();
  }

/*****************************************************************************************************************************
*************************************      LOOP Function      ****************************************************************
*****************************************************************************************************************************/

void loop(){
  // receive packets, get ready for callbacks
  ether.packetLoop(ether.packetReceive());
}
```

Figure 104: Arduino terminal Code Part 4

6. Electrical Wiring

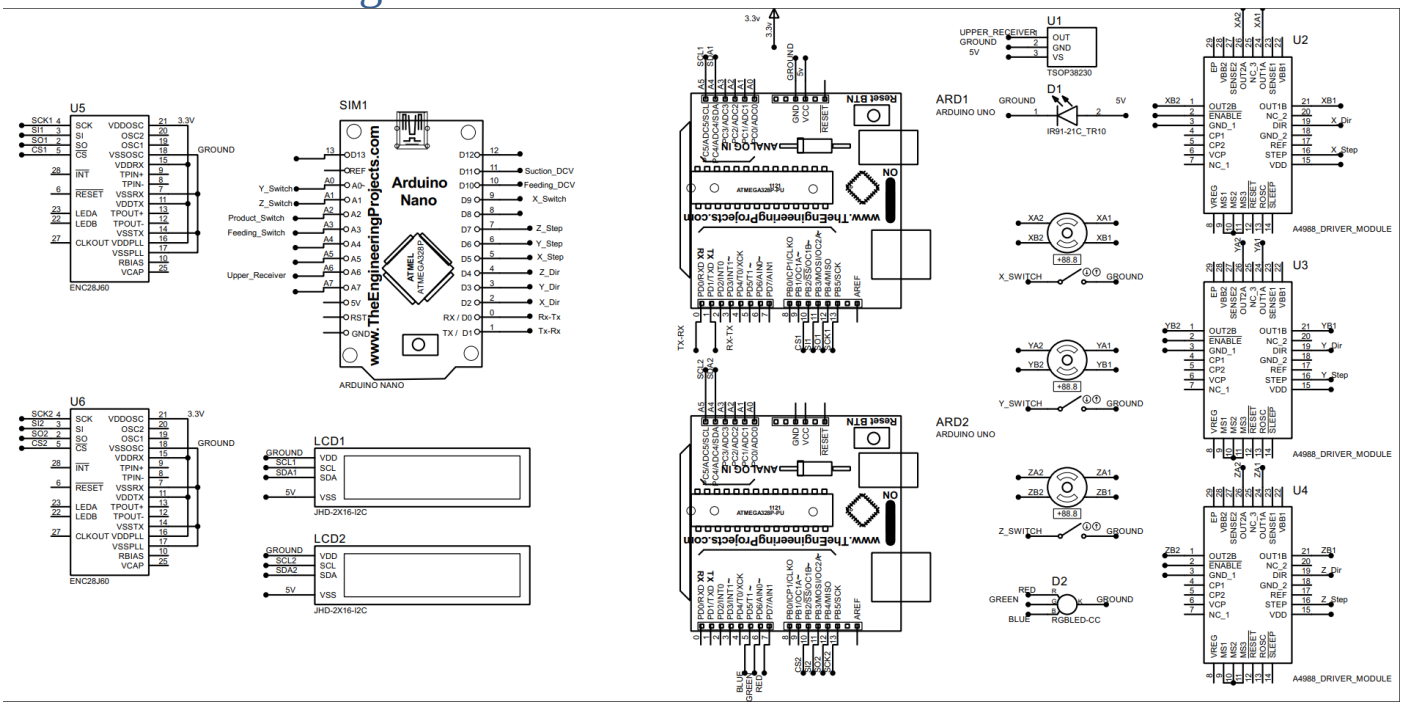


Figure 105: Electrical Wiring

7. Video Link

[MCT331_Final_Project - Google Drive]