MARMARA UNIVERSITY

FACULTY OF ENGINEERING

SEMI ACTIVE STIFFNESS CONTROL OF A 2 D.O.F. PLANAR MECHANISM

Uğur ATALMA, Onur EROL

GRADUATION PROJECT REPORT

Department of Mechanical Engineering

Supervisor

Dr. Ömer Haluk BAYRAKTAR

ISTANBUL, 2020

MARMARA UNIVERSITY

FACULTY OF ENGINEERING

Semi Active Stiffness Control of A 2 D.O.F. Planar Mechanism

by

Uğur ATALMA, Onur EROL

(150411052), (150411039)

January, 2020, Istanbul

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

BACHELOR OF SCIENCE

AT

MARMARA UNIVERSITY

Signature of Author(s)…………………………………………………………………….

Department of Mechanical Engineering

Certified By…………………………………………………………………………….

Project Supervisor, Department of Mechanical Engineering

Accepted By……………………………………………………………………………

Head of the Department of Mechanical Engineering

ACKNOWLEDGEMENT

First of all, we would like to thank our supervisor Dr. Ömer Haluk BAYRAKTAR for valuable guidance and advice on preparing this thesis also giving us moral and kindly support.

Secondly, we would like to thank Hermetal Machinery for the raw material support and letting us using their equipment’s.

Last of all, beside Dr. Ömer Haluk BAYRAKTAR we would like to thank all the professors and research assistants for the education they gave to us and ability to think and solve problems as an engineer.

January, 2020 Uğur ATALMA, Onur EROL

CONTENTS

ACKNOWLEDGEMENT i
CONTENTS ii
ÖZET iii
ABSTRACT iv
SYSMBOLS v
ABBREVIATIONS vi
LIST OF FUGURES vii
LIST OF TABLES ix
1. INTRODUCTION 1
2. MECHANICAL COMPONENTS / MANUFACTURING PROCESS 3
    2.1. Components 3
        2.1.1. Main plate 3
        2.1.2. Carrier plate 3
        2.1.3. Vibration plate and clamps 5
        2.1.4. M14 pivot bolt 7
        2.1.5. M12x1 shaft and nut 9
        2.1.6. Spring 11
    2.2. Design 13
3. ELECTRONIC COMPONENTS / SETUP AND CODING 16
    3.1. Electronic Components 16
        3.1.1. Arduino Uno 17
        3.1.2. Stepper motor 18
        3.1.3. Stepper motor driver 19
        3.1.4. Vibration DC motor 20
        3.1.5. Accelerometer 21
        3.1.6. Joystick module 21
        3.1.7. MOSFET 22
        3.1.8. LCD screen 23
        3.1.9. Accelerometer 23
        3.1.10. Potentiometer 24
    3.2. Electronic Setup and Wiring 25
    3.3. Coding 29
        3.3.1. Accelerometer and vibration motor codes 30
        3.3.2. Stepper motor with joystick module and LCD screen coding 32
4. RESULTS AND CONCLUSIONS 38
REFERENCES 44
APPENDICES 45

ÖZET

Temel olarak, bu projede, yayları kullanarak iç kuvvetleri degiştirmek suretiyle titreşim plakasının titreşimini azaltmaya çalıştık. İlk olarak, tasarımımıza ve bu tasarımın etkili ve uygun bir şekilde nasıl üretileceğine karar verdik. Daha sonrasında elektronik bileşenlerimize karar verdik ve sistemin montajı üzerinde çalıştık. Bu raporda tüm mekanik ve elektronik bileşenlerimiz listelenmiş ve açıklanmıştır. Bu bileşenler gelecekteki çalışmalar için değiştirilebilir.

Bu raporda öncelikle üretim sürecinden, daha sonrasında ise elektronik tasarım ve kodlamadan bahsettik. Çalışmalarımızın ana mantığı, titreşim plakasını 3 yay kullanarak çekme kuvveti ile kontrol etmektir. Yaylardaki gerilim, titreşim plakasına 3 farklı noktadan çekme kuvveti uygular. Rotasyon kademeli motorlarla yayı hangi miktarda çekeceğimizi kontrol ettik.

Donanım ve yazılım olarak Arduino ve platformunu kullandık. Titreşim plakasına takılan bir ivme ölçerle Arduino üzerinden verileri gözlemledik. Bu projede amacımız titreşimin genliğini kontrol etmek ve azaltmaktır.

Bu projede kapsamında başta mekanik tasarımımız olmak üzere sıfırdan bir sistem imal ettik. Mekanik ve elektronik bileşenlerimizi bu projenin sonunda bölümümüze bırakıyoruz. Gelecek dönem öğrencilerinin bu sistemi geliştirip, üzerinde yeni fikirler bulacağını umuyoruz.

ABSTRACT

Basically, in this project we are trying to passively control the stiffness at the tip point of a parallel plate mechanism. Firstly, we have decided our design and how to produce that design with an effective and affordable way. Then we decided our electronic components and studied on system assembly. In this report all our mechanical and electronic components are listed and explained. These components may be modified for future studies.

In this report, firstly we mentioned about manufacturing process and after that electronic design and coding. Main operating logic of our study is to control stiffness at the center point of a vibration plate via pulling force with using 3 spring. Tension on the springs applies the pulling force to vibration plate from 3 different points. We control how much we are going to pull the spring with stepper motors.

As a hardware and software, we have used Arduino and its platform. With an accelerometer which attached to the vibration plate we can observe and gain data through Arduino. Our aim is control and reduce amplitude of vibration.

On this project we have built something from scratch and we left our mechanical and electronic components to our department. We hope future semester students will improve and find new ideas over this study.

SYSMBOLS

a	**:** Acceleration
F	**:** Force
g	**:** Acceleration of gravity
k	**:** Stiffness
mm	**:** Millimeter
ms	**:** Millisecond
t	**:** Thickness
V	**:** Volt
W	**:** Watt

ABBREVIATIONS

A	**:** Ampere
AC	**:** Alternating Current
CNC	**:** Computer Numerical Control
DC	**:** Direct Current
D.O.F.	**:** Degree of Freedom
ERM	**:** Eccentric Mass Vibration Motor
HRC	**:** Rockwell C Hardness
IDE	**:** Integrated Development Environment
I/O	**:** Input/output
LED	**:** Light-emitting Diode
LCD	**:** Liquid Crystal Display
MOSFET	**:** Metal Oxide Semiconductor Field Effect Transistor
PCB	**:** Printed Circuit Board
rpm	**:** Revolution per minute
USB	**:** Universal Serial Bus

LIST OF FIGURES

PAGE

Figure 1.1. Basic diagram of the mechanism 2

Figure 1.2. Assembled top view of the mechanism....................2

Figure 2.1. Material analysis for main plate 4

Figure 2.2. Material analysis for carrier plate....................5

Figure 2.3. Material analysis for vibration plate and clamps 6

Figure 2.4. CNC operation on the vibration plate 6

Figure 2.5. Basic principle of wire erosion method 7

Figure 2.6. The raw material that clamps are manufactured with wire erosion method ..7

Figure 2.7. Material analysis for M14 pivot blot before nickel electroplating 8

Figure 2.8. Material analysis for M14 pivot blot after nickel electroplating 8

Figure 2.9. Basic electrical circuit for electroplating 9

Figure 2.10. Material analysis for M12x1 shaft 10

Figure 2.11. Material analysis for M12x1 square nut....................10

Figure 2.12. Process information of black phosphate coating....................11

Figure 2.13. Manufacturing of spring on the turning lathe machine....................12

Figure 2.14. Heat treatment process on springs....................12

Figure 2.15. Technical drawing of main plate....................13

Figure 2.16. Bottom surface of the carrier plate....................14

Figure 2.17. Top surface of the carrier plate 14

Figure 2.18. Assemble of the stepper motor, M12x1 shaft and square nut, the hook 15

Figure 2.19. Thrust bearing 15

Figure 3.1. Arduino Uno with Atmega328 microcontroller....................17

Figure 3.2. Nema 17HS3001 stepper motor....................19

Figure 3.3. A4988 stepper motor driver 20

Figure 3.4. WRK-450SH vibration DC motor 20

Figure 3.5. ADXL345 accelerometer board 21

Figure 3.6. Joystick module...22

Figure 3.7. MOSFET 22

Figure 3.8. 16x2 LCD screen...23

Figure 3.9 200W 12V power supply 24

Figure 3.10. 3 terminal potentiometer 24

Figure 3.11. 5x7 cm perfboard 25

Figure 3.12. Accelerometer and vibration DC motor with potentiometer wiring setup.26

Figure 3.13. Stepper motors with LCD panel, drivers and joystick wiring setup 27

Figure 3.14. Arduino IDE interface..29

Figure 4.1. Stiffness and potential energy graph for springs...39

Figure 4.2. Acceleration vs time graph when there is no force acting on plate.............41

Figure 4.3. Acceleration vs time graph when F=16.16 N...42

Figure 4.4. Acceleration vs time graph when there is no vibration...............................43

LIST OF TABLES

PAGE

Table 2.1. List of mechanical components 3

Table 3.1. List of electronic components...16

Table 3.2. Electronic component classification for Figure 3.12. and Figure 3.13..........28

1. INTRODUCTION

Stiffness is the capacity of a mechanical system to sustain loads without excessive changes of its geometry. It is one of the most important design criteria for mechanical components and mechanisms, and every material has its own stiffness characteristic. That characteristic defined by physical and chemical properties of the material. The stiffness (k), of a body is a measure of the of the resistance offered by an elastic body to deformation. For an elastic body with a single D.O.F. (degree of freedom), for example stretching or compression of a spring, the stiffness is defined as;

$$k = F/x \tag{1.1}$$

where k stands for the force, x stands for the displacement (the change in length of a stretched or compressed spring).

Since we try to control the vibrating component on this project, we need to define vibration as well. Vibration is a mechanical phenomenon where oscillations occur about an equilibrium point. The oscillations may be periodic, such as the motion of a pendulum or random, such as the movement of a tire on a gravel road. Vibration can be desirable: for example, the reed in a woodwind instrument or harmonica, the motion of a tuning fork or the cone of a loudspeaker. In many cases however vibration is undesirable, wasting energy and creating unwanted sound. For example, the vibrational motions of engines, electric motors, or any mechanical devices in operation are typically unwanted. In this project we tried to isolate the vibration caused by the vibration motor. We are trying to control the vibration and tried to reduce its amplitude with using the springs and stepper motors.

In this project report we are going to talk about how we chose the materials during our project process. It was important to think about what type of material we are going to use. We have mostly used aluminum instead of steel because it’s so much lighter and resistant to corrosion. Also, we were supposed to select the electronic components that everybody can obtain easily besides should have been user friendly. We will also talk about why did we chose certain sensors and parts in the system.

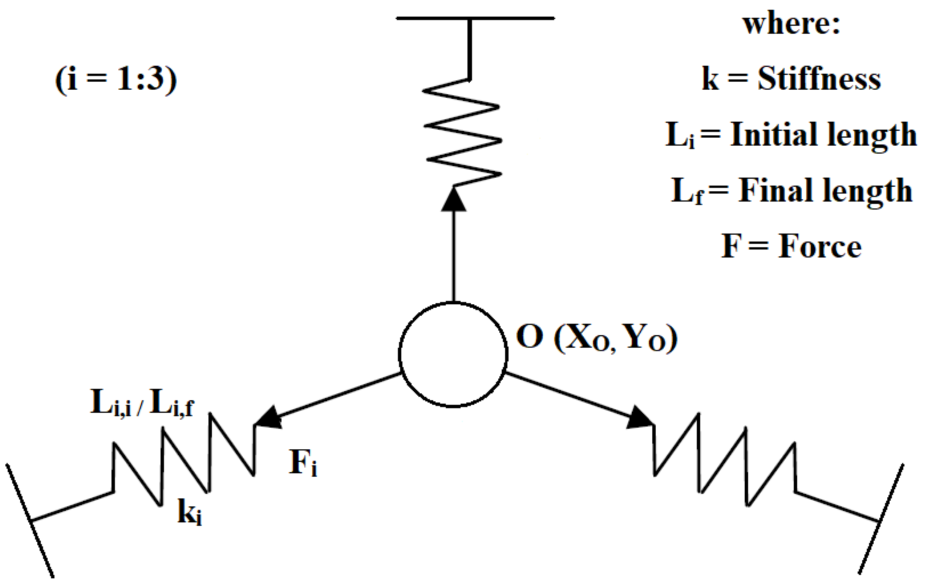


Figure 1.1. Basic diagram of the mechanism

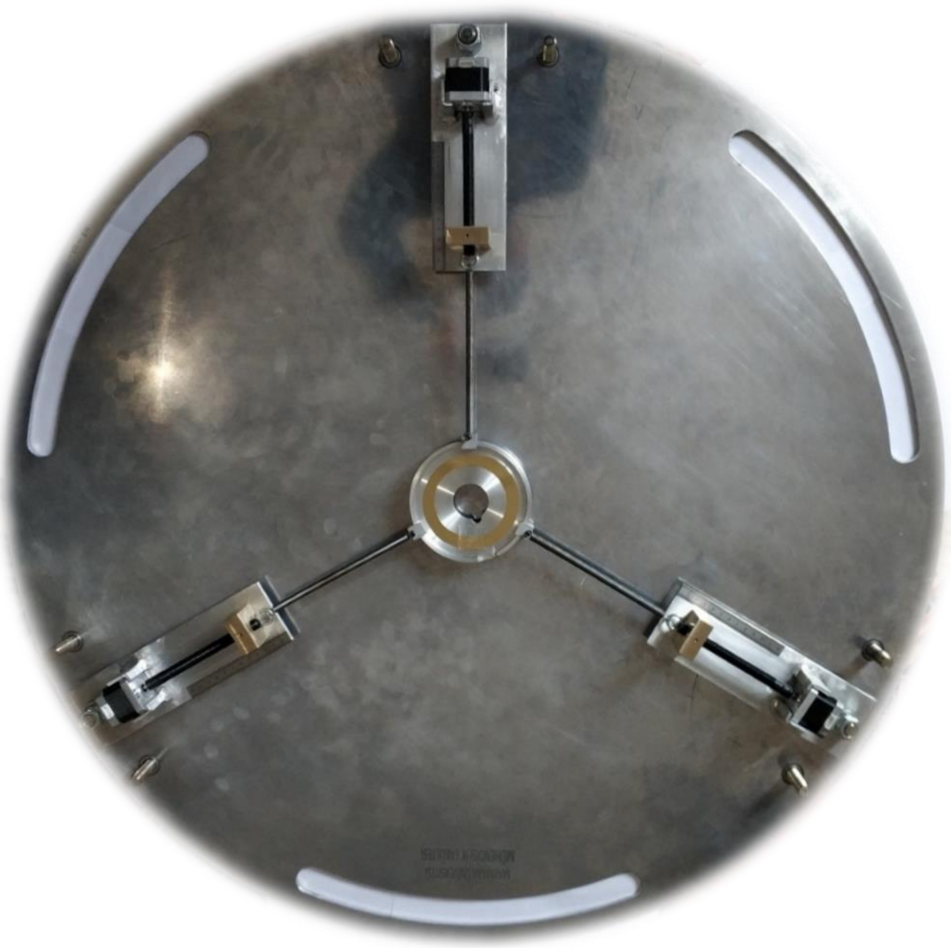


Figure 1.2. Assembled top view of the mechanism

2. MECHANICAL COMPONENTS / MANUFACTURING PROCESS

At this chapter we are going to introduce you our components and manufacturing methods. While designing and choosing materials for our components we try to ensure that future semester students could work on that design. In order to do that we protect all components from corrosion.

2.1. Components

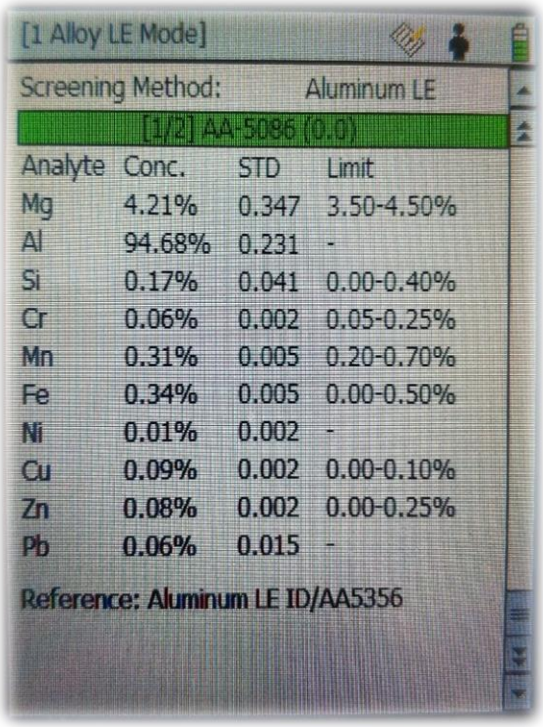
Table 2.1. List of mechanical components

#	Items	Material	Quantities
1	Main Plate	Aluminum – 5086	1
2	Carrier Plate	Aluminum – 5754	3
3	Vibration Plate	Aluminum – 6082	1
4	Clamp	Aluminum – 6082	3
5	M14 Pivot Bolt	Steel – 1020	9
6	M12x1 Shaft	Steel – 1020	3
7	M12x1 Square Nut	Aluminum Bronze – C632	3
8	6901 Bearing	Stainless Steel	9
9	Ball Transfer Unit	Stainless Steel	6
10	Thrust Bearing	Stainless Steel	1
11	Hook	Steel – 1020	3
12	Spring	Spring Steel	3

2.1.1. Main plate

Main plate is one of the most important component in our design. All moving elements are placed on the main plate so it should be big enough for the assembly. In order to do that we decide main plate diameter for 1000 millimeter (mm). On the other hand, it should be light in weight so we determine our thickness (t) for 6 millimeter (mm). Also, we choose aluminum as a material because it's density much less than steels' even it is much costly than steel ($\rho_{aluminum} = 2.73 \frac{gram}{\text{centimeter}^3}$, $\rho_{steel} = 7.86 \frac{gram}{\text{centimeter}^3}$). Besides that, it is resistance to corrosion.

While producing that component in stated dimensions (Ø1000x6 mm) we use laser cutting method because it is not compatible for turning lathe machine. Material analysis for the main plate is given at the figure below.



[1 Alloy LE Mode]

Screening Method: Aluminum LE

[1/2] AA-5086 (0.0)

Analyte	Conc.	STD	Limit
Mg	4.21%	0.347	3.50-4.50%
Al	94.68%	0.231	-
Si	0.17%	0.041	0.00-0.40%
Cr	0.06%	0.002	0.05-0.25%
Mn	0.31%	0.005	0.20-0.70%
Fe	0.34%	0.005	0.00-0.50%
Ni	0.01%	0.002	-
Cu	0.09%	0.002	0.00-0.10%
Zn	0.08%	0.002	0.00-0.25%
Pb	0.06%	0.015	-

Reference: Aluminum LE ID/AA5356

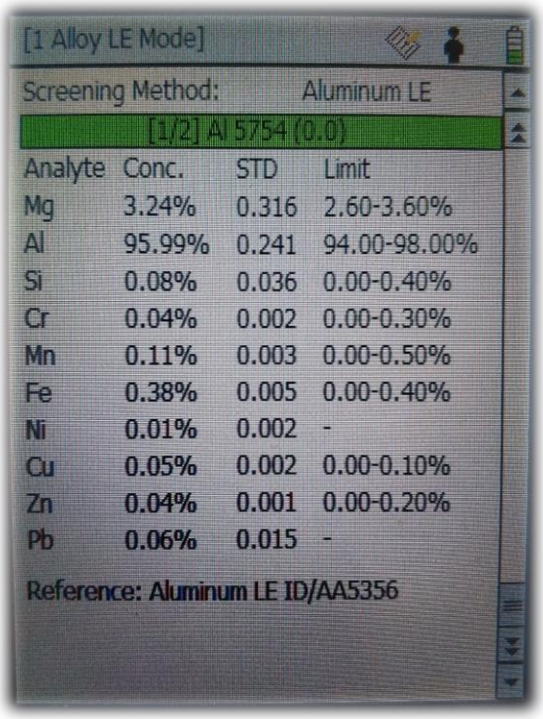
Figure 2.1. Material analysis for main plate

Laser cutting method

Laser cutting is a technology that uses a laser to slice materials. While typically used for industrial manufacturing applications, it is also starting to be used by schools, small businesses, and hobbyists. Laser cutting works by directing the output of a high-power laser most commonly through optics. The laser optics and CNC (computer numerical control) are used to direct the material or the laser beam generated. A commercial laser for cutting materials involved a motion control system to follow a CNC or G-code of the pattern to be cut onto the material. The focused laser beam is directed at the material, which then either melts, burns, vaporizes away, or is blown away by a jet of gas, leaving an edge with a high-quality surface finish. Industrial laser cutters are used to cut flat-sheet material as well as structural and piping materials. [1]

2.1.2. Carrier plate

Carrier plate is the platform that stepper motor is placed. This platform also carries the M12x1 square nut and the movement of the M12x1 square nut takes place on this platform. We choose aluminum as a material again for its light weight and resistance to corrosion. Material analysis for carrier plate is given at the figure below.



[1 Alloy LE Mode]

Screening Method: Aluminum LE

[1/2] Al 5754 (0.0)

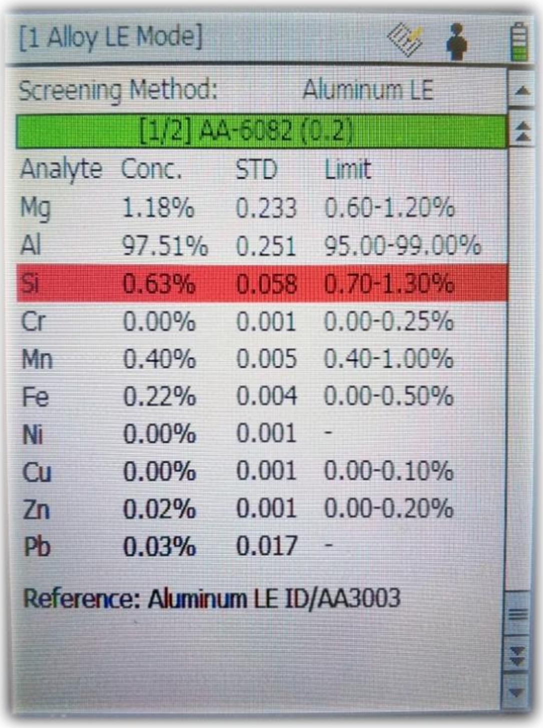
Analyte	Conc.	STD	Limit
Mg	3.24%	0.316	2.60-3.60%
Al	95.99%	0.241	94.00-98.00%
Si	0.08%	0.036	0.00-0.40%
Cr	0.04%	0.002	0.00-0.30%
Mn	0.11%	0.003	0.00-0.50%
Fe	0.38%	0.005	0.00-0.40%
Ni	0.01%	0.002	-
Cu	0.05%	0.002	0.00-0.10%
Zn	0.04%	0.001	0.00-0.20%
Pb	0.06%	0.015	-

Reference: Aluminum LE ID/AA5356

Figure 2.2. Material analysis for carrier plate

2.1.3. Vibration plate and clamps

Vibration plate is the plate that vibration DC (direct current) motor and accelerometer placed on. Clamps are the joint member for the vibration plate and the spring. We choose aluminum as a material for these two components again for same reasons. Production of vibration plate, firstly we use manual turning lathe machine and then CNC turning lathe machine. On the CNC turning lathe machine, we have machined the channel where thrust bearing will sit. CNC codes and their explanations are given at the Appendix I. For the production of clamps, we use wire erosion method because of precision machining. Material analysis for that components are given at the figure below.


[1 Alloy LE Mode]

Screening Method: Aluminum LE

[1/2] AA-6082 (0.2)

Analyte	Conc.	STD	Limit
Mg	1.18%	0.233	0.60-1.20%
Al	97.51%	0.251	95.00-99.00%
Si	0.63%	0.058	0.70-1.30%
Cr	0.00%	0.001	0.00-0.25%
Mn	0.40%	0.005	0.40-1.00%
Fe	0.22%	0.004	0.00-0.50%
Ni	0.00%	0.001	-
Cu	0.00%	0.001	0.00-0.10%
Zn	0.02%	0.001	0.00-0.20%
Pb	0.03%	0.017	-

Reference: Aluminum LE ID/AA3003

Figure 2.3. Material analysis for vibration plate and clamps

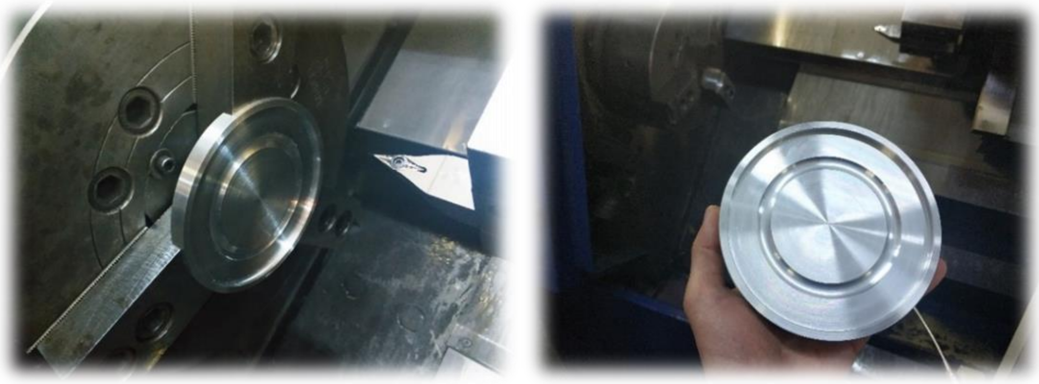


Figure 2.4. CNC operation on the vibration plate

Wire erosion method

Wire erosion is a precision engineering process where a work piece is cut through with a strand of wire. It creates electrical sparks between the wire and the work piece and these electrical sparks cut the work piece. It’s a cost-effective method that has incredible accuracy on thicker component parts. Wire cutting machines can cut complex shapes even in tough materials. Basic working principle of laser erosion method is given at the figure below. [2]

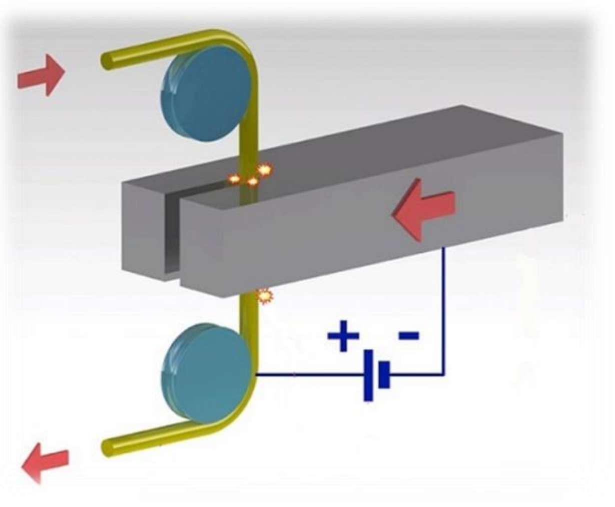


Figure 2.5. Basic working principle of wire erosion method

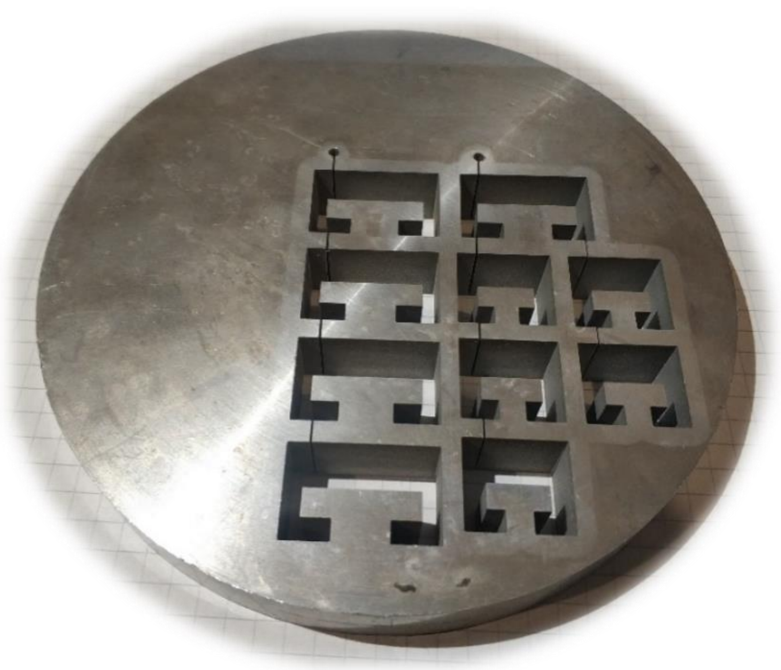
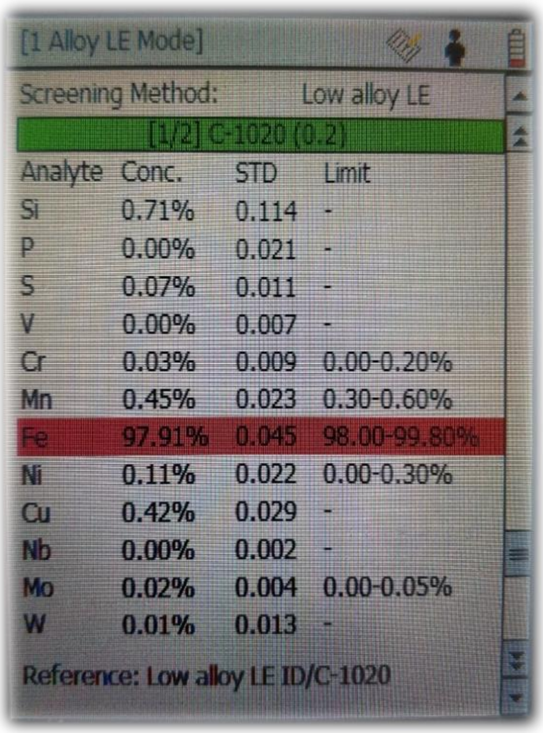


Figure 2.6. The raw material that clamps are manufactured with wire erosion method

2.1.4. M14 pivot bolt

That component is used for fasten the carrier plate to the main plate. We use 1020 steel for that component but for the rust protection we make nickel electroplating after the production. Material analysis before and after nickel electroplating are given at the figures below.

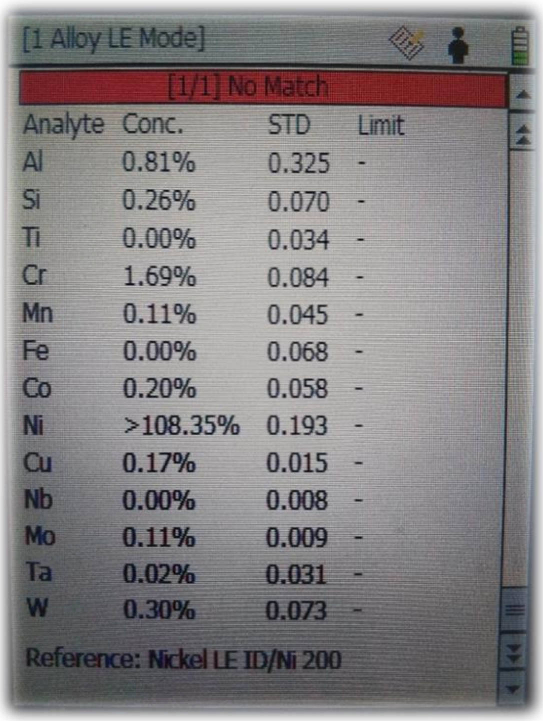


[1 Alloy LE Mode]

Screening Method:	Low alloy LE		
	[1/2] C-1020 (0.2)		
Analyte	Conc.	STD	Limit
Si	0.71%	0.114	-
P	0.00%	0.021	-
S	0.07%	0.011	-
V	0.00%	0.007	-
Cr	0.03%	0.009	0.00-0.20%
Mn	0.45%	0.023	0.30-0.60%
Fe	97.91%	0.045	98.00-99.80%
Ni	0.11%	0.022	0.00-0.30%
Cu	0.42%	0.029	-
Nb	0.00%	0.002	-
Mo	0.02%	0.004	0.00-0.05%
W	0.01%	0.013	-

Reference: Low alloy LE ID/C-1020

Figure 2.7. Material analysis for M14 pivot blot before nickel electroplating



[1 Alloy LE Mode]

[1/1] No Match

Analyte	Conc.	STD	Limit
Al	0.81%	0.325	-
Si	0.26%	0.070	-
Ti	0.00%	0.034	-
Cr	1.69%	0.084	-
Mn	0.11%	0.045	-
Fe	0.00%	0.068	-
Co	0.20%	0.058	-
Ni	>108.35%	0.193	-
Cu	0.17%	0.015	-
Nb	0.00%	0.008	-
Mo	0.11%	0.009	-
Ta	0.02%	0.031	-
W	0.30%	0.073	-

Reference: Nickel LE ID/Ni 200

Figure 2.8. Material analysis for M14 pivot blot after nickel electroplating

Nickel electroplating

Electroplating is a relatively easy and inexpensive way of depositing thin and thick films on conductive substrates without damaging the substrates. Nickel electroplating is a technique of electroplating a thin layer of nickel into a metal object. The nickel layer can be decorative and provide corrosion resistance. Nickel is considered useful for electroplating metal because it provides superior ductility, corrosion resistance, and hardness. Electro nickel plating can also improve a product's brightness and external appearance. Different nickel-plating chemicals incorporated into the process deliver anything from a semi-bright and fully bright cosmetic effect, to matte, pearl, or satin finishes. [3]

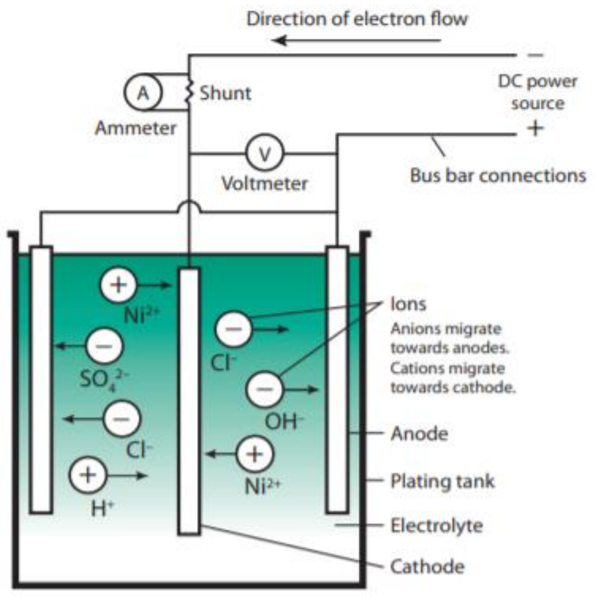
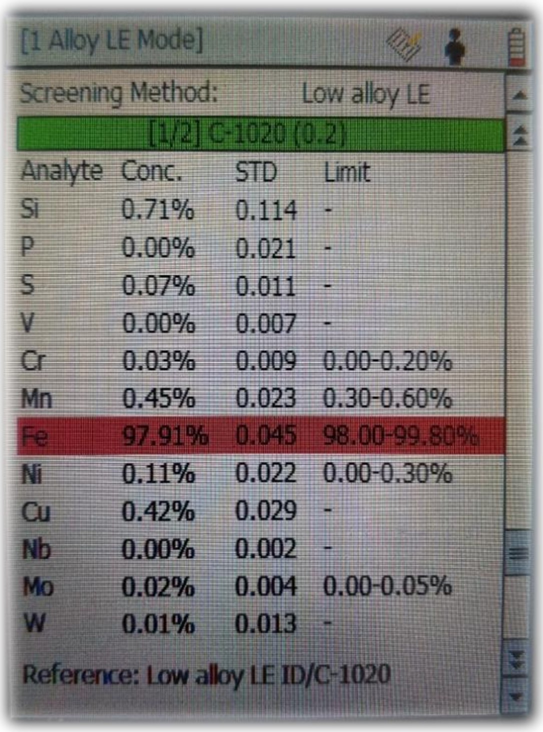


Figure 2.9. Basic electrical circuit for electroplating

2.1.5. M12x1 shaft and square nut

M12x1 shaft and square nut components have an important duty on our design. Both components tie the stepper motor to the spring. We use 1020 steel as a material for shaft and aluminum bronze as a material for square nut. After producing of the shaft, we applied black phosphate coating process for rust protection. The reason for choosing aluminum bronze as a material for square nut is visual aspect. For both components' material analysis are given at the figures below.



[1 Alloy LE Mode]

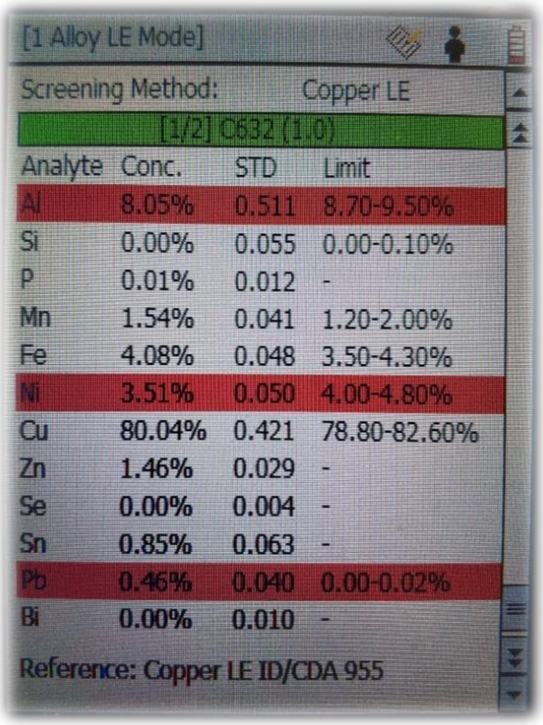
Screening Method: Low alloy LE

[1/2] C-1020 (0.2)

Analyte	Conc.	STD	Limit
Si	0.71%	0.114	-
P	0.00%	0.021	-
S	0.07%	0.011	-
V	0.00%	0.007	-
Cr	0.03%	0.009	0.00-0.20%
Mn	0.45%	0.023	0.30-0.60%
Fe	97.91%	0.045	98.00-99.80%
Ni	0.11%	0.022	0.00-0.30%
Cu	0.42%	0.029	-
Nb	0.00%	0.002	-
Mo	0.02%	0.004	0.00-0.05%
W	0.01%	0.013	-

Reference: Low alloy LE ID/C-1020

Figure 2.10. Material analysis for M12x1 shaft



[1 Alloy LE Mode]

Screening Method: Copper LE

[1/2] C632 (1.0)

Analyte	Conc.	STD	Limit
Al	8.05%	0.511	8.70-9.50%
Si	0.00%	0.055	0.00-0.10%
P	0.01%	0.012	-
Mn	1.54%	0.041	1.20-2.00%
Fe	4.08%	0.048	3.50-4.30%
Ni	3.51%	0.050	4.00-4.80%
Cu	80.04%	0.421	78.80-82.60%
Zn	1.46%	0.029	-
Se	0.00%	0.004	-
Sn	0.85%	0.063	-
Pb	0.46%	0.040	0.00-0.02%
Bi	0.00%	0.010	-

Reference: Copper LE ID/CDA 955

Figure 2.11. Material analysis for M12x1 square nut

Black phosphate coating

Black phosphate coating products are made in order to cover steel and casting surfaces such as machine gears, gun parts with black color at high temperatures. Following this process, the parts are quickly fed into the oil baths, so that the steel surfaces provide maximum corrosion resistance. Blaphoric U 22 is used to coat iron, steel and casting surfaces with black phosphate layer. The bath runs at boiling temperature. The processing time is between 5 and 15 minutes. Parts must be cleaned in degreasing and rust removing baths before taking black phosphate treatment. After the oxidation process, the parts are rinsed and taken to the oil baths. [4]

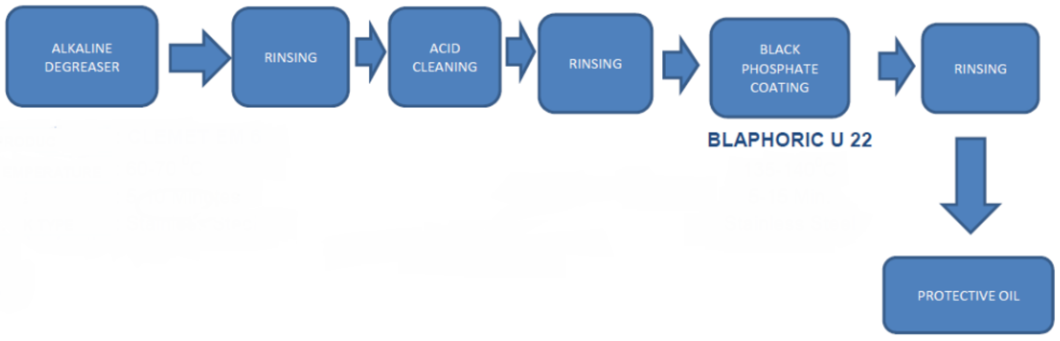


Figure 2.12. Process information of black phosphate coating

2.1.6. Spring

Spring is an elastic mechanical component that stores mechanical energy. The are many types of springs like tension spring, compression spring, torsion spring, constant spring, variable spring etc. We use tension spring in our design and we choose 1.4568 spring steel as a material. 1.4568 spring steel feature is very high and resistant to heat and corrosion. Although it can reach high surface tension with heat treatments, it is highly resistant to fatigue. After the heat treatment process our springs hardness reached 50 HRC (Rockwell C Hardness). Production processes of springs are given at the figures below.

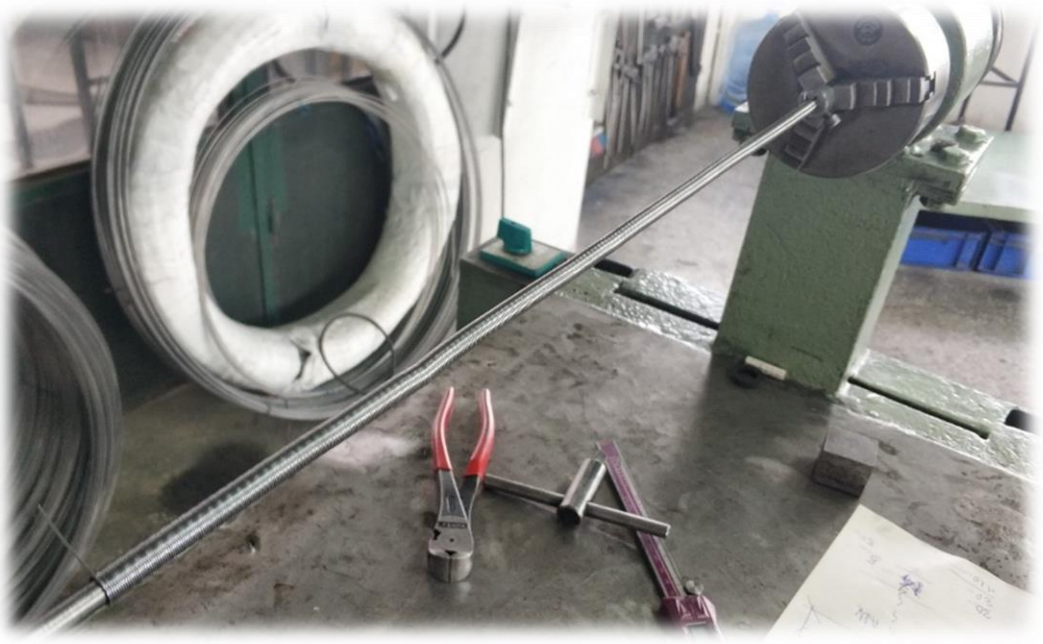


Figure 2.13. Manufacturing of spring on the turning lathe machine

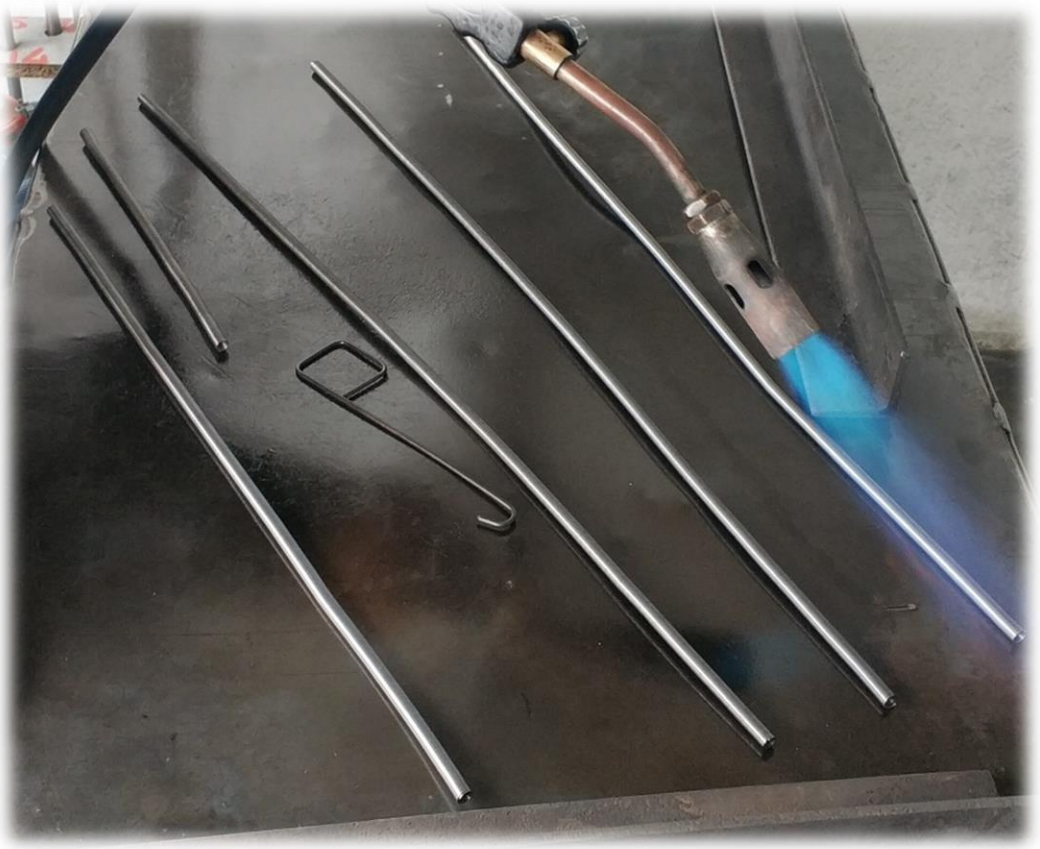


Figure 2.14. Heat treatment process on springs

2.2. Design

While we assemble our design, we also use some stockpile mechanical components. These are 6901 bearings, ball transfer units, thrust bearing and hooks. For the design, first step was deciding the main plate dimensions. As we mentioned on the 2.1. Main plate part our dimensions is Ø1000x6 mm. Technical drawing of main plate is given at the figure below.

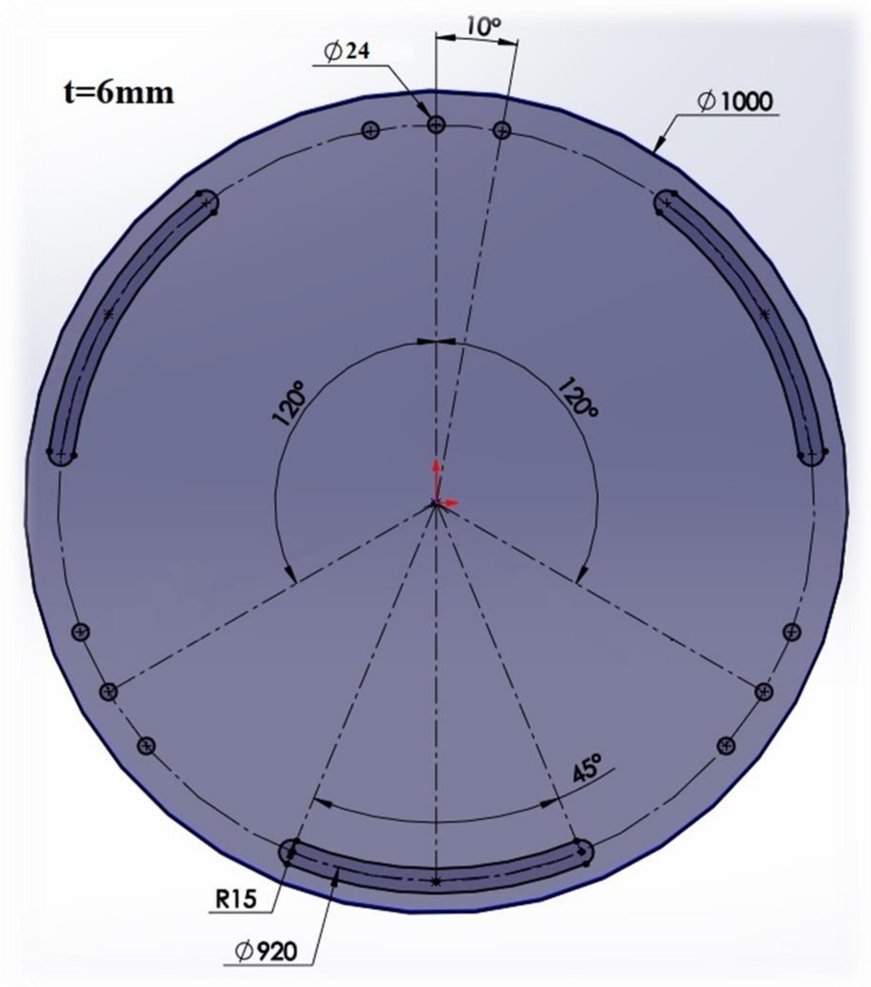


Figure 2.15. Technical drawing of main plate

Across the Ø920 mm line there are 3 pieces of slots for carrying the plate easily and there are 9 pieces of Ø24 mm holes for penetrating the 6901 bearings. The reason for choosing 6901 bearing is its width is 6 mm. So, when we penetrate the bearings into the main plate there isn't any overlap between them. On the inner diameter of the bearings we penetrate M14 pivot bolts. Within that mounting these 3 components became as a one.

We mounted the carrier plates to the main plate with the help of M14 pivot blots. Carrier plates has an important duty on our system. While our system is working, stepper motors on the carrier plates transfer their rotations to the M12x1 shaft and then with that rotation M12x1 square nuts moves along in a straight line on the carrier plates. In order to ensure the balance in the z axis for the carrier plate, we have installed 2 pieces of ball transfer units under each of them. We mounted M12x1 shaft to the stepper motor shaft via using 2 pieces of M3 stay blots. With that connection we can transmit the rotation of the stepper motors as a pulling force on the springs.

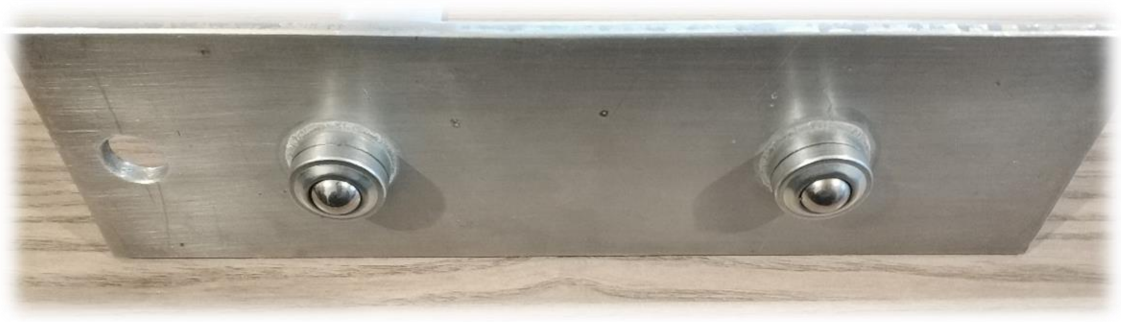


Figure 2.16. Bottom surface of the carrier plate

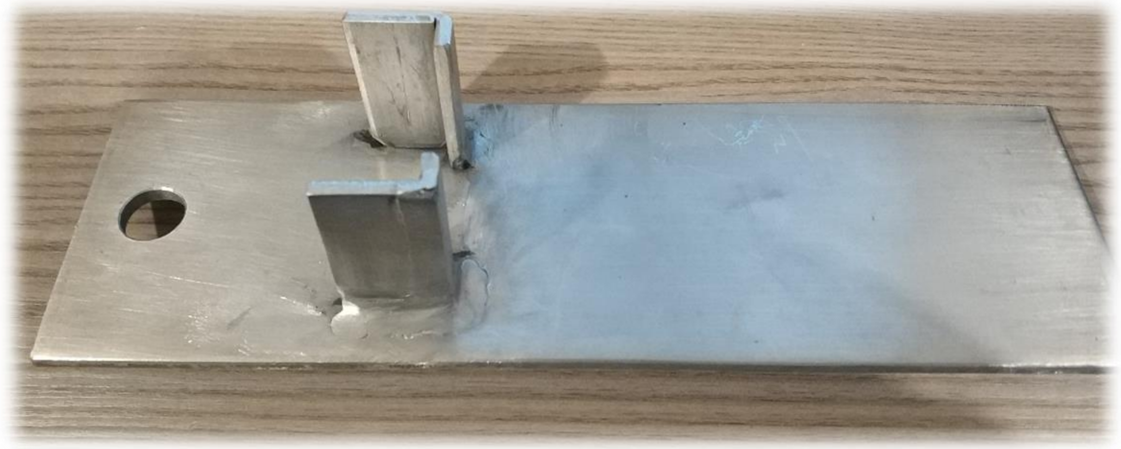


Figure 2.17. Top surface of the carrier plate

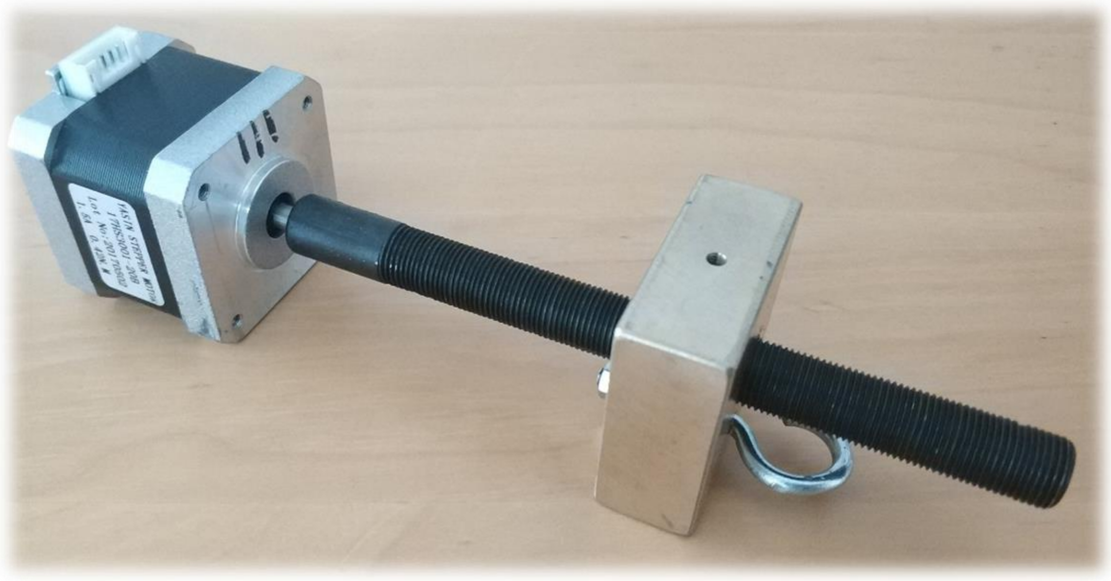


Figure 2.18. Assemble of the stepper motor, M12x1 shaft and square nut, the hook

With the help of the hook seen at the Figure 2.18. we can attach M12x1 square nuts to the vibration plate via using the spring. While one tip of the spring attached to the hook other tip attached to the clamps so that's how we made spring connection in our system.

Normally M12 thread pitch is 1.75 mm but we are using special thread which thread pitch is 1 mm so we equalized stepper motors' rotation number to distance travelled in mm for square nuts. [5]

As we mentioned earlier at 2.1.3. Vibration plate and clamps part, we placed vibration DC motor and accelerometer on the top of the vibration plate. To vibrate smoothly on the main plate, we use thrust bearing between them.

Figure 2.19. Thrust bearing

3. ELECTRONIC COMPONENTS / SETUP AND CODING

For control the mechanism and stepper motors we have used is a hardware called Arduino Uno which is simply a circuit board that has microcontroller and processor with a software in it accepts C and C++ in it as programming language. Arduino is an open-source electronics platform based on easy to use hardware and software. Being the open-source platform makes the product is open to public so people can access, design and make adjustments for your hardware and software. This makes the Arduino is so user-friendly and also, it’s much easier to find helpful tools crated by the users. It was the best choice for us to use for this system because it is cheap and easy to learn also can be accessed community crated projects and codes from the internet. Many people whose interested in robotics as a hobby or people who want to create something new with sensors, Arduino is great tool for start. In this chapter we are going to explain how the system is controlled by the codes and also the electronic system we have designed for our project.

3.1. Electronic Components

We have several materials and electronics part. All electronic parts that we used in the system are given at the table below.

Table 3.1. List of Electronic Components

#	Items	Model No	Quantities
1	Arduino Uno	CH340	2
2	Stepper Motor	17HS3001	3
3	Stepper Motor Driver	A4988	3
4	Vibration DC Motor	WRK-450SH	1
5	Accelerometer	ADXL345	1
6	Joystick Module	PS2	1
7	MOSFET	14A400W PWM	1
8	LCD Screen	2x16 I2C	1
9	12 V Power Supply	WPS-12V-1.6A-200W	1
10	Potentiometer	WH148	1

3.1.1. Arduino Uno

Arduino is an open hardware development board that can be used by hobbyists and makers to design and build devices that interact with the real world. While Arduino refers to a specific type of board design, it can also be used to refer to a company which manufactures a specific implementation of these boards, and is typically also used to describe the community around compatible boards made by other people or companies which function in a similar way. Arduinos contain a number of different parts and interfaces together on a single circuit board. The design has changed through the years, and some variations include other parts as well. But a basic board shown in Figure 3.1. with detailed name of components if desired to check. [6]

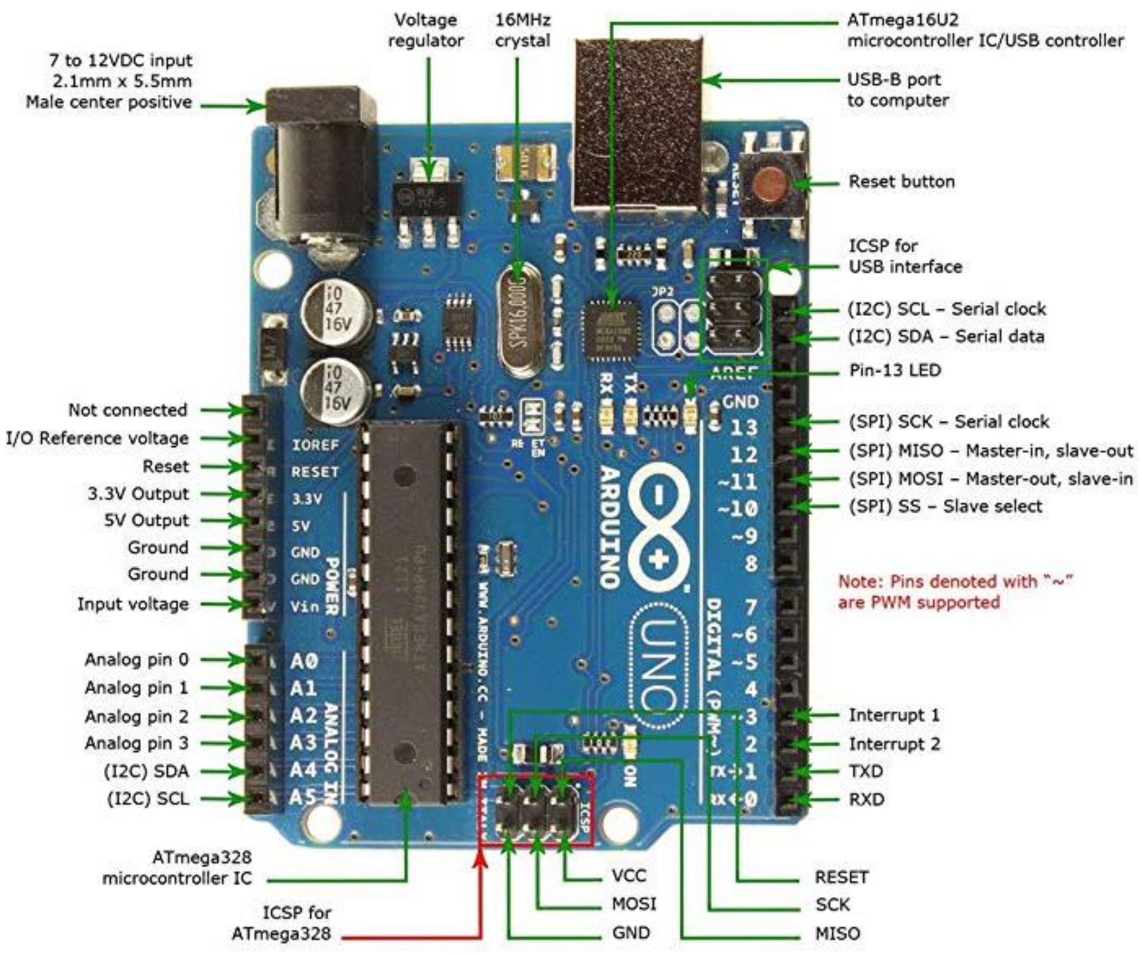


Figure 3.1. Arduino Uno with Atmega328 microcontroller

A number of pins, which are used to connect with various components to use with the Arduino. These pins come in two varieties:

- **Digital pins**, which can read and write a single state on or off. Most Arduinos have 14 digital I/O (Input/Output) pins,
- **Analog pins,** which can read a range of values, and are useful for more accurate control of the mechanism or sensors. Most Arduinos have six of these analog pins.

A power connector, which provides power to both the device itself, and provides a low voltage which can power connected components like LEDs (Light-emitting Diode) and various sensors, provided their power needs are reasonably low. The power connector can connect to either an AC (Alternating Current) adapter or a small battery. A microcontroller, the primary chip, which allows you to program the Arduino in order for it to be able to execute commands and make decisions based on various input. The exact chip varies depending on what type of Arduino you buy, but they are generally Atmel controllers, usually an ATmega8, ATmega168, ATmega328, ATmega1280, or ATmega2560. The differences between these chips are subtle, but the biggest difference a beginner will notice is the different amounts of onboard memory. A serial connector, which on most newer boards is implemented through a standard USB (Universal Serial Bus) port. This connector allows you to communicate to the board from your computer, as well as load new programs onto the device. Often times Arduinos can also be powered through the USB port, removing the need for a separate power connection. In our design we needed 2 Arduino Uno. One of the Arduinos works with stepper motors and LED screen with joystick module. Other one is working with accelerometer and vibration DC motor. Technical specification and schematic of Arduino Uno are given at Appendix II.A and Appendix II.B.

3.1.2. Stepper motor

The stepper motor is an electromagnetic device that converts digital pulses into mechanical shaft rotation. Advantages of step motors are low cost, high reliability, high torque at low speeds and a simple, rugged construction that operates in almost any environment.

Stepper motor can be a good choice whenever controlled movement is required. They can be used to advantage in applications where you need to control rotation angle, speed, position and synchronism. Stepper motor is a brushless DC motor that rotates in steps. This is very useful because it can be precisely positioned without any feedback sensor, which represents an open-loop controller. The stepper motor consists of a rotor that is generally a permanent magnet and it is surrounded by the windings of the stator. As we activate the windings step by step in a particular order and let a current flow through them, they will magnetize the stator and make electromagnetic poles respectively that will cause propulsion to the motor. So that the basic working principle of the stepper motors.
On each step of our stepper motors rotates 1 step out of 200 steps for full 360^0 rotation. It means that it rotates about 1.8^0 on each step. This also shows how precise of stepper motors on accurate rotational required systems.

Figure 3.2. Nema 17HS3001 stepper motor

The reason we have used the stepper motor is for achieving desired turns. This also provide us the good control of how much we need to rotate the stepper motor. Anytime on running the code we can stop and change the rotation properties like angular velocity and direction of the motor with a joystick module. Datasheet of our Nema 17HS3001 stepper motor is given at Appendix III.

3.1.3. Stepper motor driver

A stepper motor driver is an electronic device that is used to drive the stepper motor. By itself it usually does nothing. It must be used together with a controller like we have used one A4988. There are a lot of different types of stepper motor drivers but in general all do the same thing, drive stepper motors.

We needed 3 of these stepper motor drivers because each one will drive the one stepper motor. They are basically connected to each stepper motor and they also connected to the Arduino Uno.

Figure 3.3. A4988 stepper motor driver

3.1.4. Vibration DC motor

The simplest and most popular type of vibration motor is known as an ERM (Eccentric rotating mass vibration motor). It is a standard DC motor with an off-center load attached to the shaft. When the motor is powered and rotates, the unbalanced mass crates a centripetal force. If we attached the motor to an object, that force acts on the object and causes it to displace. As DC motors turn very fast, a full circle of displacement can happen well 10 to 100 times a second. This fast and repeated displacement is what we feel as vibrations. This is how a vibration DC motor works on necessary designs or environments. In this project we have used WRK-450SH model named product to vibrate the vibration plate. Data sheet of that vibration DC motor is given at Appendix V.

Figure 3.4. WRK-450SH vibration DC motor

3.1.5. Accelerometer

An accelerometer is an electromechanical device used to measure acceleration forces. Such forces may be static, like the continuous force of gravity or, as is the case with many mobile devices, dynamic to sense movement or vibrations. Acceleration is the measurement of the change in velocity.

$$a = dv/dt \tag{3.1}$$

Where a is the accelerationan, dv is infinitly small amount of change in velocity and dt represents the infintly small change in time. They measure in meters per second squared or in G-forces. A single G-force for us here on planet Earth is equivalent to 9.81 m/s^2 but this does vary slightly with elevation (and will be a different value on different planets due to variations in gravitational pull). Accelerometers are useful for sensing vibrations in systems or for orientation applications. In this project we have used the model ADXL345 which is cheap and useful.

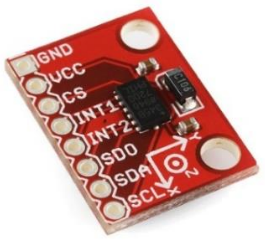


Figure 3.5. ADXL345 accelerometer board

3.1.6. Joystick module

The basic idea of a joystick is to translate the movement of a plastic stick into electronic information a computer can process. Joysticks are used in all kinds of machines, including F-15 fighter jets, backhoes and wheelchairs. In this project we have used the joystick module as a controller for rotating the stepper motors backwards and forwards. The joystick is similar to two potentiometers connected together, one for the vertical movement (Y-axis) and other for the horizontal movement (X-axis). In our design we have used only (Y-axis) because we needed only forward and backward directions for the stepper motors.

Since it acts like a potentiometer it can also read the values variable in its range on desired directions. There is also one more function of the joystick which is work as a single click button on the (Z-axis). We also used that function to call back the stepper motors to its initial positions.

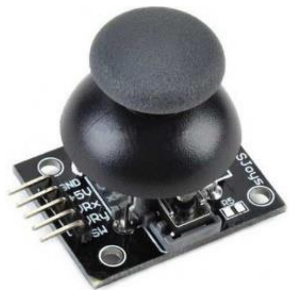


Figure 3.6. Joystick module

3.1.7. MOSFET

The MOSFET (Metal Oxide Semiconductor Field Effect Transistor) is a semiconductor device which is widely used for switching and amplifying electronic signals in the electronic devices. The MOSFET is a core of integrated circuit and it can be designed and fabricated in a single chip because of these very small sizes. The aim of the MOSFET is to be able to control the voltage and current flow between the source and drain. It works almost as a switch. It was required to use in our mechanism with the vibration DC motor. It works as well as without MOSFET but this time there is a possibility of vibration DC motor can start working even if the potentiometer is off so MOSFET is required to drive a DC motor.

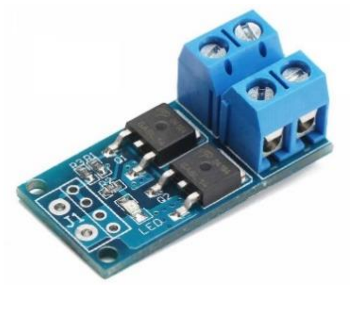


Figure 3.7. MOSFET

3.1.8. LCD screen

For monitoring the revolution of the stepper motors, we have used the 16x2 LCD (liquid crystal display) screen. The reason for named like 16x2 is that LCD has 2 lines and can display 16 characters per line. Therefore, 16x2 LCD screen can show 32 characters at once. The LCD screen can work on 4-bit mode or 8-bit mode. For the project we have used 4-bit mode because the characters we are going to type on the screen is only a couple letters.

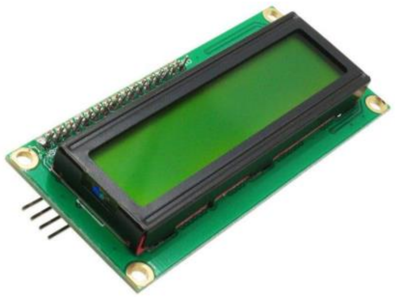


Figure 3.8. 16x2 LCD screen

3.1.9. Power supply

Power supply is an electrical device that supplies electric power. The primary function of a power supply is to convert electric current from a source to the correct voltage, current, and frequency. Power supplies are sometimes referred to as electric power converters. Some power supplies are separate standalone pieces of equipment, while others are built into the load appliances that they power. Examples of the latter include power supplies found in desktop computers and consumer electronics devices. Other functions that power supplies may perform include limiting the current drawn by the load to safe levels, shutting off the current in the event of an electrical fault, power conditioning to prevent electronic noise or voltage surges on the input from reaching the load and storing energy.

For this project we were able to obtain a 12 V (Volt) 1.6A (Ampere) and 200W (Watt) power supply to feed our electrical component in our design. Because of there were 3 stepper motors with also a DC vibration motor it requires more voltage than a USB cable support through the Arduino. Arduino supplies only 5V through his circuit board to its power pins with USB cable which is not enough for our system electrical requirement.

Figure 3.9. 200W 12V power supply

3.1.10. Potentiometer

A potentiometer is a manually adjustable variable resistor with 3 terminals. Two terminals are connected to both ends of a resistive element and the third terminal connects to a sliding contact, called a wiper, moving over the resistive element. The position of the wiper determines the output voltage of the potentiometer. The potentiometer essentially functions as a variable voltage divider. The resistive element can be seen as two resistors in series (potentiometer resistance), where the wiper position determines the resistance ratio of the first resistor to the second resistor. A potentiometer is also commonly known as a potmeter or pot. The most common form of potmeter is the single turn rotary potmeter. This type of pot is often used in audio volume control (logarithmic taper) as well as many other applications. The potentiometer is used in our design as a controller of the rpm (revolution per minute) of the vibration DC motor.

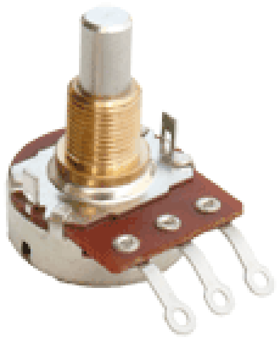


Figure 3.10. 3 terminal potentiometer

3.2. Electronic Setup and Wiring

In this chapter we talk about how the setup is made. For assembly of the wiring system, we have had a lot of help from the Arduino community because there is a lot of simple projects for each component. Electronic setup has two steps. First one is making the connections of the components which we are going to go into the detail in this chapter. Second one is the coding part also which we will be talking about in the next title.

There are a lot of wire is used in the project since we needed to connect each port for stepper motors to communicate with the Arduino while it getting data from the joystick and also giving output to the LCD screen. Since we are using plenty of wires and using most of the ports on the Arduino board, it helps a lot to use perfboard which is shown in Figure 3.11. Perfboards are useful boards are suitable for solder and after brazing it we can use it for holding the wires on the board.

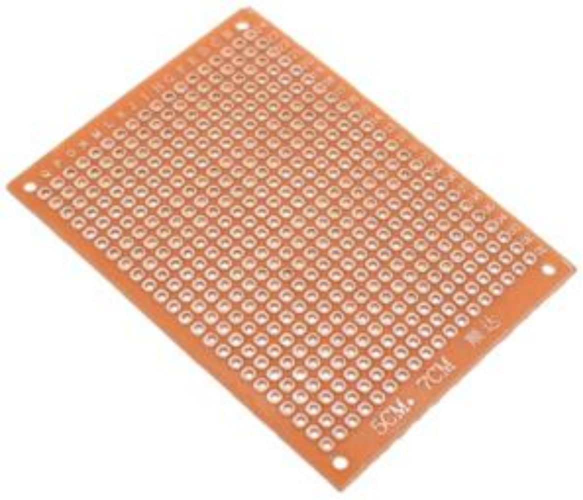


Figure 3.11. 5x7 cm perfboard

In the Figure 3.12. and Figure 3.13. the all wiring and setup is shown for the 2 different Arduino used. Figure 3.12. is setup for accelerometer and vibration DC motor. Figure 3.13. shows setup and wiring for the stepper motors with LCD screen and joystick module. We have used the software called Fritzing to crate the scheme of the system.

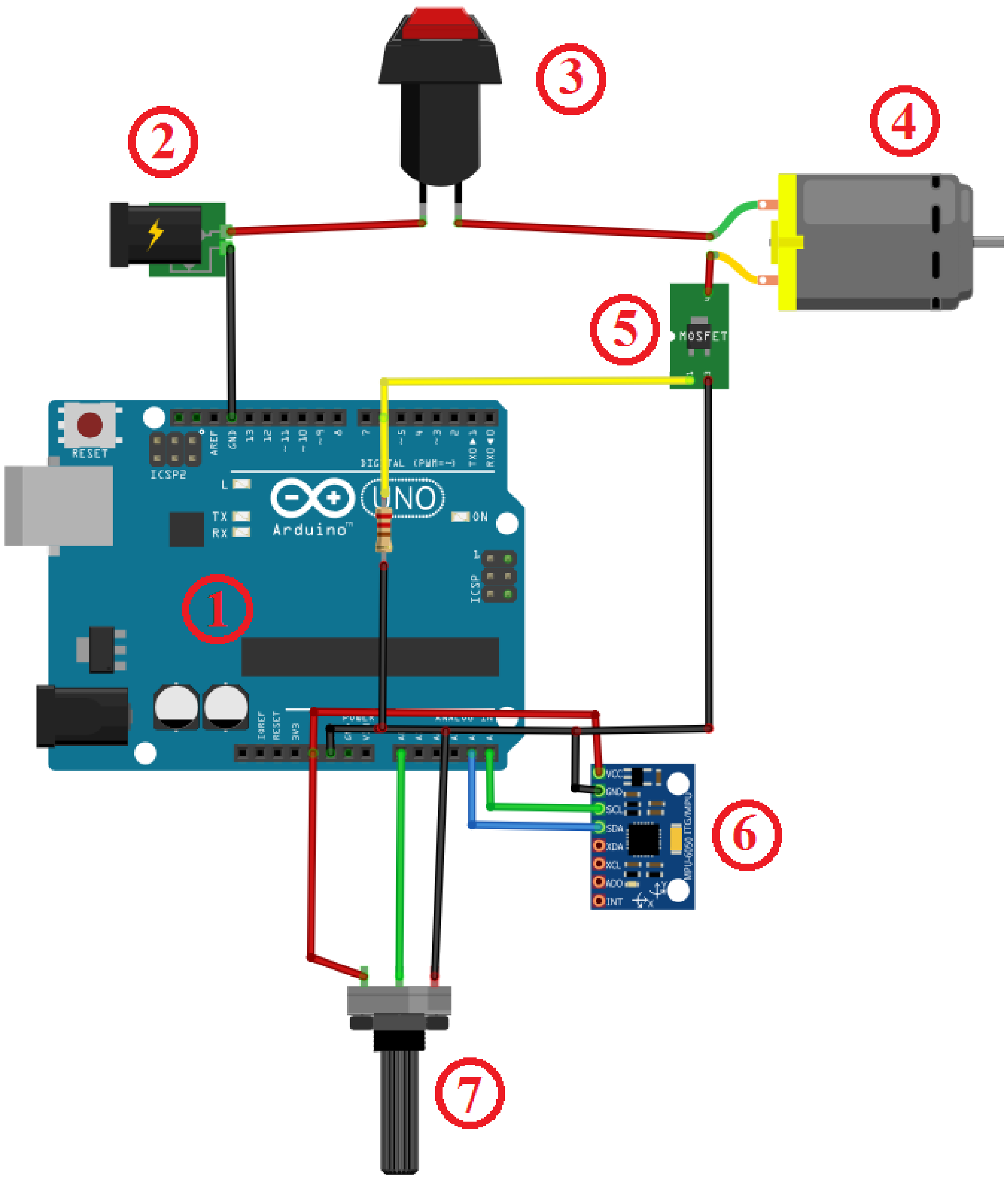


Figure 3.12. Accelerometer and vibration DC motor with potentiometer wiring setup

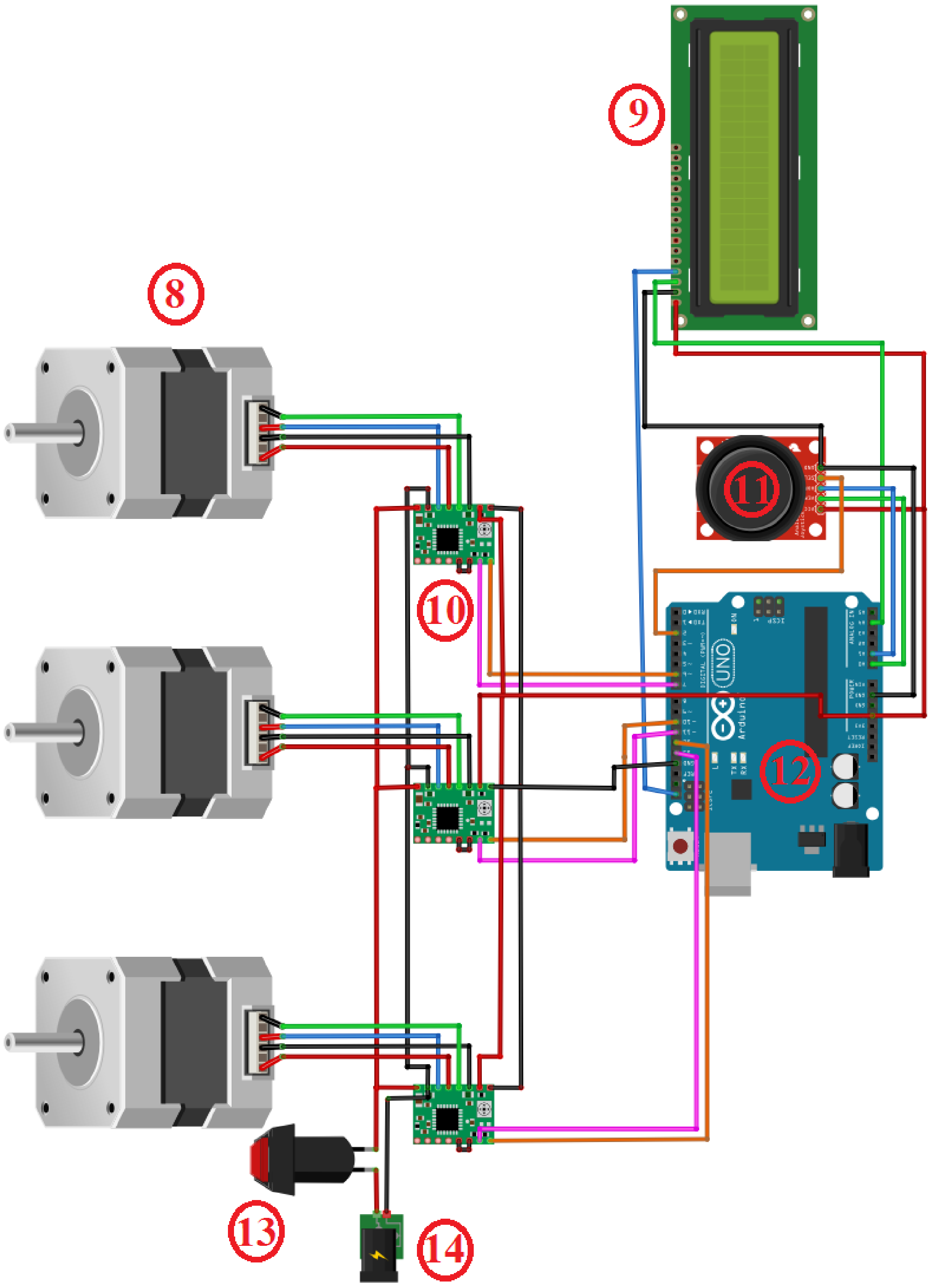


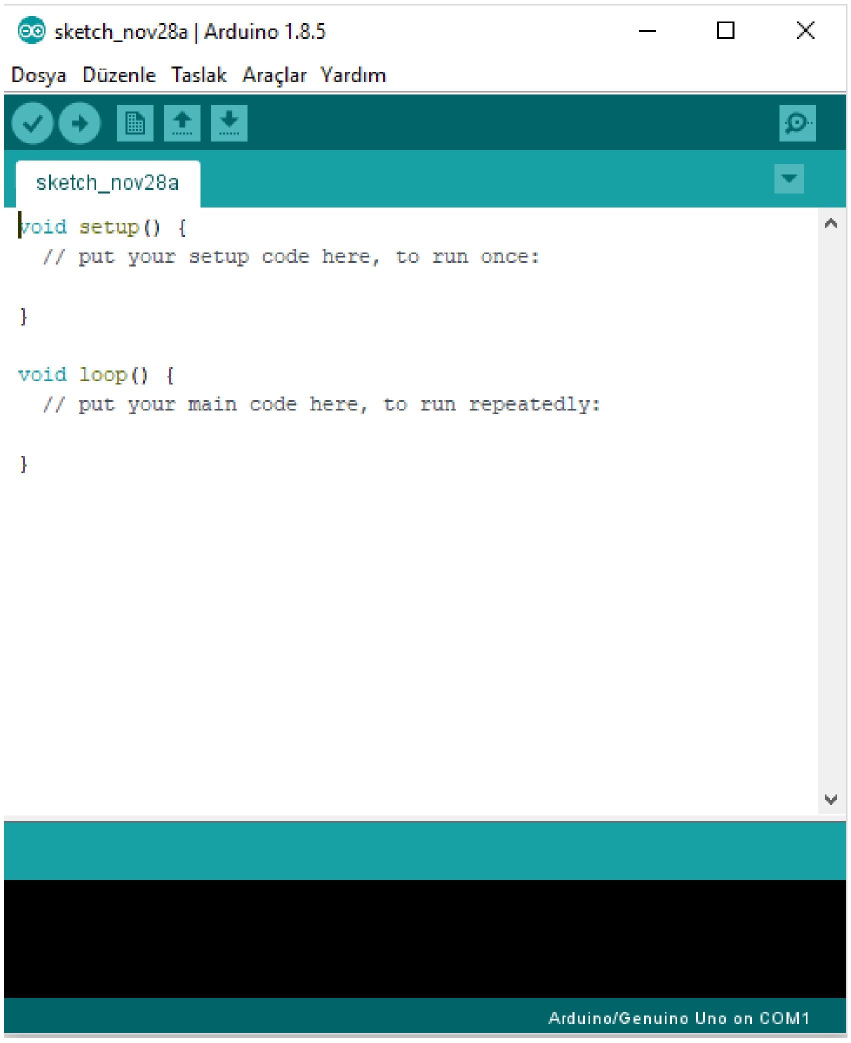
Figure 3.13. Stepper motors with drivers, LCD screen and joystick wiring setup

Table 3.2. Electronic component classification for Figure 3.12. and Figure 3.13.

#	**Items**
1, 12	Arduino Uno
2, 14	Power input
3, 13	Power switch
4	Vibration DC motor
5	MOSFET
6	Accelerometer
7	Potentiometer
8	Stepper motors
9	16x2 LCD screen
10	Stepper motor driver
11	Joystick module

As we can see from the Figures 3.12. and 3.13. we have 2 Arduino Uno platform. The reason we need to use two different PCB (Printed Circuit Board) is there was problems with the heat generation of the stepper motor drivers. Also, when we try to read the accelerometer values from the computer it was being shown with error. So, we have solved the problem with dividing the system in 2 different Arduino platform. They are still using the same power supply but on the software side of the system everything was much easier since we wrote 2 different codes for each loop.

Wiring the circuit boards also was a long time to figured out. Even though we had a lot help from the sample figure of each device, since we have around 12 components it took time to manage the wire and power connections. It’s important also to where to plug in the cables in the system because each pin connector should be defined in the code section. Arduino has its own software and code interface called IDE (Integrated Development Environment). It can be obtained from its official website and free to use. In the IDE platform we basically giving the commands to Arduino to create a communication between electrical components and Arduino board. Codes can be uploaded to Arduino via USB cable. Until you want to upload a new code the last uploaded one stays there and works unless upload a new code or click the rest button on the Arduino board as shown in Figure 3.1.


sketch_nov28a | Arduino 1.8.5
Dosya Düzenle Taslak Araçlar Yardım
sketch_nov28a
Arduino/Genuino Uno on COM1

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 3.14. Arduino IDE interface

3.3. Coding

As we can see in the Figure 3.14. there is 2 main part of the coding for communicate the Arduino with the electrical components. First one is void setup, which is the main setup as its name. Void setup codes make the system ready for to go. This part is being ran only one time before the void loop part and it stops. It basically prepares the system and configure the part however we determined the system initials. Void setup part is runs in the beginning of the job and it continues with the void loop part. Void loop section is the place where we put the codes as we desired for our system which will be continuously work until we give a new command to stop. It's a loop and works forever until we shut down the system. Actually, our main code is in the void loop section. [7, 8]

There are also libraries we have to use before the void setup. For each sensor or electronic component, we use it has its own libraries made by the manufacturers. We have to describe them into the system on our code so our parts supposed to work properly. We can see our acceleration and vibration codes in section 3.3.1.

3.3.1. Accelerometer and vibration motor codes

Void setup and adding libraries

```
//To make the I2C communication we add the wire.h library and other necessary
libraries.

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>

int motor_Pin=5,Motor_Speed=0;  //Motor is connected to pin number 6

Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);

void setup() {                              // put your setup code here, to run once:

          Serial.begin(9600);              //starting the monitoring.
          if(!accel.begin())
                    {

          Serial.println("No ADXL345 detected, check your wiring!");
                                            // if acceleration sensor not begin gives error.
          while(1);
           }
          accel.setDataRate(ADXL345_DATARATE_200_HZ);
          accel.setRange(ADXL345_RANGE_16_G);
          pinMode(motor_Pin, OUTPUT); // motor pin selected as an output
          }
```

What we did in the above codes is basically we defined and added the libraries to code for make the accelerometer works which are provided by the manufacturer of the sensors. Also setting the motor pin to output makes us able to control the vibration motor via the potentiometer. As we rotate the potentiometer the output value will change and we will be able to adjust the rotational speed of the vibration DC motor.

Void loop for printing the acceleration values on IDE

```
void loop() {
        // put your main code here, to run repeatedly:

         //mapping the analog read between 0 and 1023 to between 0 and 40 space
        Motor_Speed=map(analogRead(A1), 0, 1023,0, 50);
        analogWrite(motor_Pin, Motor_Speed);
        //motor pin will read the motor speed

        sensors_event_t event;
        accel.getEvent(&event);

        /* Display the results (acceleration is measured in m/s^2) */
        Serial.print("X: "); Serial.print(event.acceleration.x); Serial.print("  ");
        Serial.print("Y: "); Serial.print(event.acceleration.y); Serial.print("  ");
        Serial.print("Z: "); Serial.print(event.acceleration.z);
        Serial.print("  ");Serial.println("m/s^2 ");
        delay(1);
        //the delay between each measured acceleration value in ms (millisecond)
          }
```

In the void setup part, we have set the motor speed as value of the output motor pin. The potentiometers resistance value is between 0 and 1023 on the device. With the map function we spread this value between 0 and 50 to give the rotational speed to the motor. If we increase the value of 50 to 75, motors top rotational speed increases but being less sensitive. We tried to find the optimum value with the tests we made.

Serial print code is printing the values of each data taken from the accelerometer while vibration. We can see the graph of the vibration on the Arduino IDE with the port serial screen. It shows the acceleration values on X, Y and Z axis. Delay function in the code is how many ms will the loop is going to wait after each loop is finished. This also works as a adjusting the graph reading speed.

3.3.2. Stepper motor with joystick module and LCD screen coding

Like previous code on section 3.3.1. We have libraries to add and also void setup part of the code which will run only one time during the operation and stops for adjust the system.

Void setup, adding libraries and assigning the integers

```
#define DIR1      6     //choosing the pins of direction and steps
#define STEP1     7
#define DIR2      10
#define STEP2     11
#define DIR3      12
#define STEP3     13

#include <LiquidCrystal_I2C_AvrI2C.h>    // lcd libraries including

LiquidCrystal_I2C_AvrI2C lcd(0x27,16,2);  // lcd connection options setting up

boolean dir_bool = LOW,setdir;
int button_dir=3,speed,position_reset=2;

int Position=0,Analog_Value=0,cycle=0,old_cycle=0;

void setup() {

        Serial.begin(9600);                    // serial communication begins

        pinMode(button_dir,INPUT);         // setting the button as input
        pinMode(position_reset, INPUT_PULLUP);
```

```
          pinMode(DIR1,OUTPUT);             // setting the outputs
          pinMode(STEP1,OUTPUT);
          pinMode(DIR2,OUTPUT);
          pinMode(STEP2,OUTPUT);
          pinMode(DIR3,OUTPUT);
          pinMode(STEP3,OUTPUT);

          lcd.begin();                      // begin the lcd

          lcd.clear();                      // clear the lcd
          delay(5);                         // wait for this step
          lcd.setCursor(0,0);               // set lcd cursor to initial
          lcd.print("REVOLUTION 0");        // write the position
          }
```

Each command on the programming is described what they do in the system as a comment as we can see above. We set the system for once at the start and also added the libraries for the sensors and motors work. Next thing we are going to do is determining the void loop and also there is a new function here we crated named void reset. This makes the stepper motors move to its initial position. Whenever we want to it will go back to reset function to initialize the system.

Void reset for setting the stepper motors to initial position

```
void reset(){                           // crated for resetting option
          speed=1500;                   // constant speed value at first
          while(Position!=0){
          if(Position>0){               // if position greater than 0 decrease and reverse
          Position--;
          digitalWrite(DIR1,LOW); // controlling direction with setting DIR low or high
          digitalWrite(DIR2,LOW);
          digitalWrite(DIR3,LOW);
```

```
        digitalWrite(STEP1,HIGH); // Making steps one high and one low stepper
rotate 1 step
        digitalWrite(STEP2,HIGH);
        digitalWrite(STEP3,HIGH);
        delayMicroseconds(speed);   // high speed value in delay, low speed on rotation
        digitalWrite(STEP1,LOW);
        digitalWrite(STEP2,LOW);
        digitalWrite(STEP3,LOW);
        delayMicroseconds(speed);

   }else{
        // if you position greater than 0 increase the position and
rotate reverse

   Position++;
   digitalWrite(DIR1,HIGH);       // setting DIR low and high controlling direction
   digitalWrite(DIR2,HIGH);
   digitalWrite(DIR3,HIGH);

   digitalWrite(STEP1,HIGH);      // Making steps one high and one low stepper rotate
1 step
   digitalWrite(STEP2,HIGH);
   digitalWrite(STEP3,HIGH);
   delayMicroseconds(speed);        // high speed value in delay low speed on rotation
   digitalWrite(STEP1,LOW);
   digitalWrite(STEP2,LOW);
   digitalWrite(STEP3,LOW);
   delayMicroseconds(speed);
     }
      old_cycle=Position/200;
      if(old_cycle!=cycle)
     {
      cycle = old_cycle;
      lcd.clear();                      // clear the lcd
      delay(5);                         // wait for this step
      lcd.setCursor(0,0);               // set lcd cursor to initial
      lcd.print("Revolution ");         // set the position
      lcd.print(cycle, DEC);            // write the position

      }
 }
}
```

As we can see in the codes above, we wrote the first reset function. Reset function works when we choose to reset to position to its original place. If the stepper motors moved forward rotating as we start the reset function it is going back to its initial position at constant rotational speed. Next step is setting up the void loop function which will continue until we stop. Loop functions continuously works if we don't stop them.

Void loop for rotating stepper motors with joystick module

```
void loop() {

            if(!digitalRead(position_reset))            //  if  joystick  button  clicked,  go  to
position_reset
               reset();

             Analog_Value=analogRead(A1);        // reading joystick values and setting it
to external variable

             if(Analog_Value>562){                    //prevent to joystick works at released
position also setting stepper motors direction due to location of joystick
            speed=map(Analog_Value, 562, 1023,4000, 1000);        // mapping the analog
value for desired range
```

```
 Position++;
   digitalWrite(DIR1,HIGH);        // controlling direction with setting DIR low or high
   digitalWrite(DIR2,HIGH);
   digitalWrite(DIR3,HIGH);
   digitalWrite(STEP1,HIGH);
   digitalWrite(STEP2,HIGH);
   digitalWrite(STEP3,HIGH);
   delayMicroseconds(speed);        // high speed value in delay low speed on rotation
   digitalWrite(STEP1,LOW);
   digitalWrite(STEP2,LOW);
   digitalWrite(STEP3,LOW);
   delayMicroseconds(speed);

  }

  else if(Analog_Value<462){
  speed=map(Analog_Value, 0, 462,1000, 4000);

Position--;
   digitalWrite(DIR1,LOW);
   digitalWrite(DIR2,LOW);
   digitalWrite(DIR3,LOW);
   digitalWrite(STEP1,HIGH);
   digitalWrite(STEP2,HIGH);
   digitalWrite(STEP3,HIGH);
   delayMicroseconds(speed);
   digitalWrite(STEP1,LOW);
   digitalWrite(STEP2,LOW);
   digitalWrite(STEP3,LOW);
   delayMicroseconds(speed);

  }
 if(Position<0){

          old_cycle=Position/200;
          if(old_cycle!=cycle){
          cycle = old_cycle;
          lcd.clear();                          // clear the lcd
          delay(5);                             // wait for this step
          lcd.setCursor(0,0);                  // set lcd cursor to initial
          lcd.print("REVOLUTION ");   // set position
          lcd.print(-cycle, DEC);            // write the location

          }

  }else{

           old_cycle=Position/200;

             if(old_cycle!=cycle){

            cycle =(old_cycle);
            lcd.clear();                        // clear the lcd
            delay(5);                           // wait for this step
            lcd.setCursor(0,0);                // set lcd cursor to initial
            lcd.print("REVOLUTION "); // set position
            lcd.print(-cycle, DEC);          // write the location

        }
    }
}
```

These codes also can be seen in Appendix V.A and V.B as a whole without explanations and comments.

The void loop part is where the stepper motors rotates as we move the joystick to the forward or backward. When we push the joystick to the forward stepper motors will rotate on clockwise direction and pull the square nuts and springs so pulling the springs will cause the force acting on the vibration plate and we will try to decrease the vibration of the vibration plate.

Basically, the vibration occurs on the equation of;

$$F = mr\omega^2 \tag{3.2}$$

where F is centripetal force due to rotation, m is the mass of the eccentric mass on the vibration motor and ω is the rotational speed of the shaft. When we have the equation of

$$F = ma \tag{3.3}$$

with where m mass is the total mass of the plate and vibration DC motor, a is the acceleration of the vibration plate. Because of the eccentricity on the vibration motor weight, it will cause displacement on our vibration plate. We call the acceleration of the plate on total weight is amplitude. The easiest way to varying the amplitude of the system is just changing the voltage is given to vibration DC motor. Like we do in our system via using potentiometer.

4. RESULTS AND CONCLUSIONS

After we made bunch of tests, we observed different acceleration values depend on how we locate accelerometer on the vibration plate. When we take high frame rate video, we have seen the displacement of the vibration plate is decreasing and that increases the stiffness of the vibration plate. As we rotate the vibration DC motor, the vibration plate starts to oscillate and has an amplitude of displacement. At the first stage of the test the square nuts are at the initial position. As we start to rotate stepper motors on clockwise position, the springs stretching and applying the extension force. The bolt and stepper motor pull the springs 1 mm to backward as it makes one full revolution. For the springs as it compressed or extended it stores potential energy. If we assign U as potential energy, k for the spring stiffness constant and x for the displacement [9]. We have,

$$U = \frac{1}{2}kx^2 \tag{4.1}$$

If we derivate the equation above it yields,

$$\frac{\partial U}{\partial x} = kx \tag{4.2}$$

While,

$$\frac{\partial U}{\partial x} = F \tag{4.3}$$

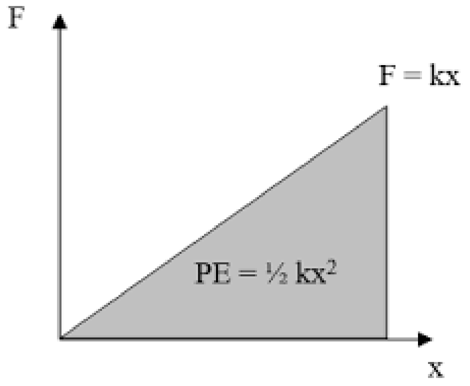


Figure 4.1. Stiffness and potential energy graph for springs

In our experiments we have 4 different sets of spring which all sets have different stiffness constant. Each of the springs are 162 mm long. It was fit to between vibrating plate and square nuts with this length. With the stiffest spring type, we have made an experiment, we hang a 5 kg (kilogram) weight on the spring and measured the displacement it made from its original unstretched length. After the weight, length of the spring was 405 mm in total means we have the displacement x is 243 mm. Since the gravitational acceleration on earth $g = 9.81\ m/s^2$, From the Newton's Law,

$$F = ma$$

$$F = (5)(9.81) = 49.05\ N$$

Since we know the stiffness equation from Eq. 1.1.

And we have the displacement x of 243 mm on the experiment,

$$49.05\ N = k\ (0.243\ m)$$

So, the stiffness value of the stiffest spring yields,

$$\boldsymbol{k = 202\ N/m}$$

In our system the M12x1 shaft which is connected to stepper motor rotates 80 revolution until the tooth of the shaft ends. That means each full rotation of the stepper motor pulls the springs 1 mm. As we rotate the stepper motor 80 revolution, the displacement will be 80 mm which is 0.08 meter. As a result of that, the force acting on the plate for each spring, since they rotate on same revolution is,

$$F = kx$$

$$F = \left(202\frac{N}{m}\right)(0.08\,m)$$

$$F = 16.16\,N$$

For each spring force F acting on the vibration plate.

On the other hand, there is also an acceleration on the vibrating plate and this creates a continuously acceleration on the accelerometer sensor. There were also the graph results of the acceleration for the sensor which we obtained from the Arduino IDE serial screen. The graph results show us 2 different status of the system. One of it is the acceleration graph on Figure 4.2. when the springs stays on initial position which means there is no force acting the plate. Second graph on Figure 4.3. shows the acceleration during a certain amount of time when we pull the springs 80 mm long which we already calculated the force F acting on the plate.

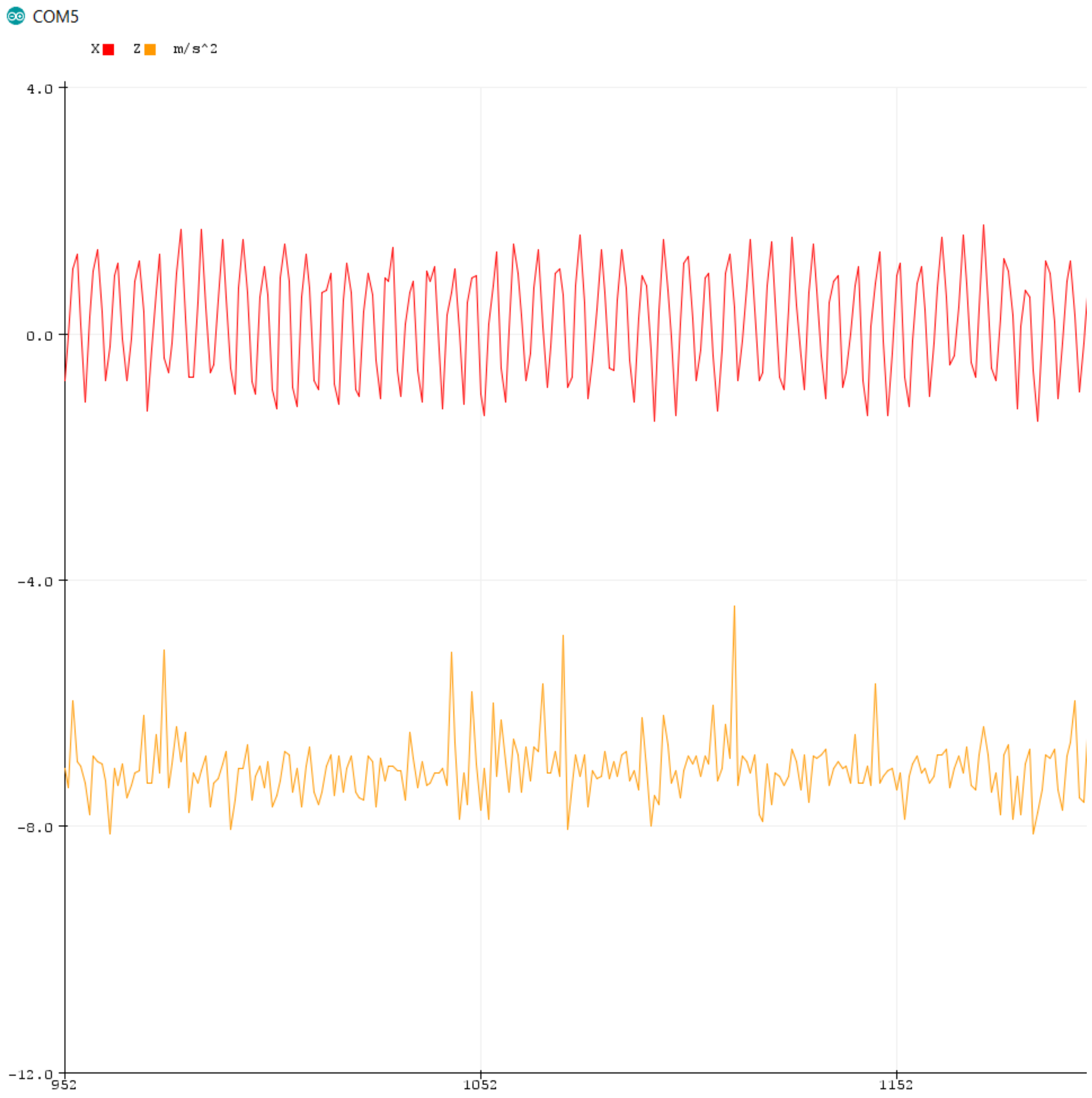


Figure 4.2. Acceleration vs time graph when there is no force acting on plate

On the Figure 4.2. graph shows only in X and Z-axis since we have a circular plate and X and Y-axis plane are on the main plane which is horizontal. All forces acting from 120^o from each side but it can be changed by using other M14 pivot bolts. X and Y values of the acceleration would be similar. The values are on m/s^2 because the library we added to code already contains this unit for acceleration. For the g values we can divide the written values with 9.81 if desired on the code. Z-axis acceleration is oscillating because there was still some amount of vibration on the Z-axis because of the system configuration. The vibration plate is not heavy enough to prevent vibration on vertical axis.

On the other hand, we have also Figure 4.3. which contains the graph of the acceleration of the system when we pull the springs 80 mm and crates the 16.16 N and on the Figure 4.4. we can see the data on the initial position without vibration DC motor not rotating.

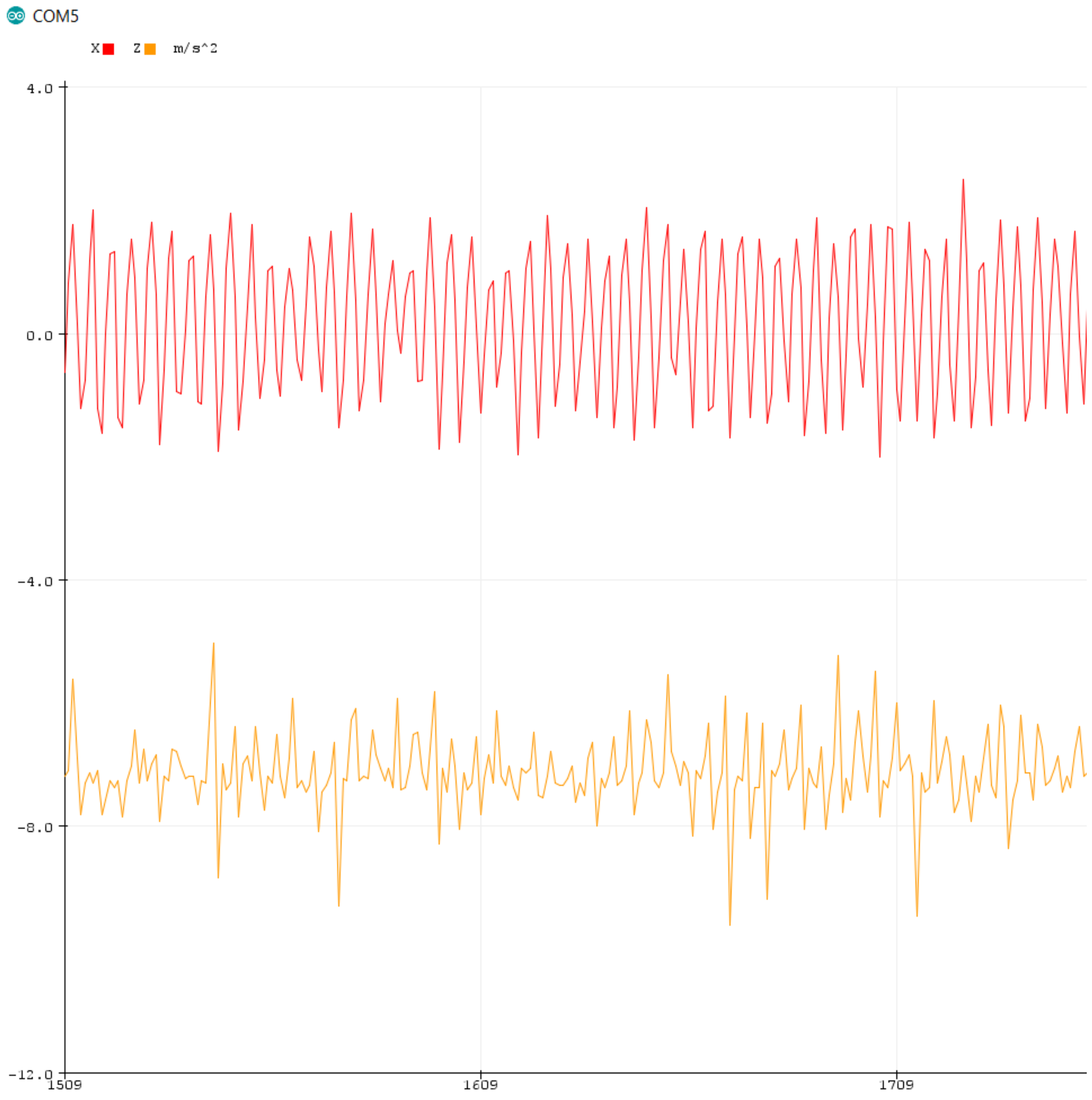


Figure 4.3. Acceleration vs time graph when F=16.16 N

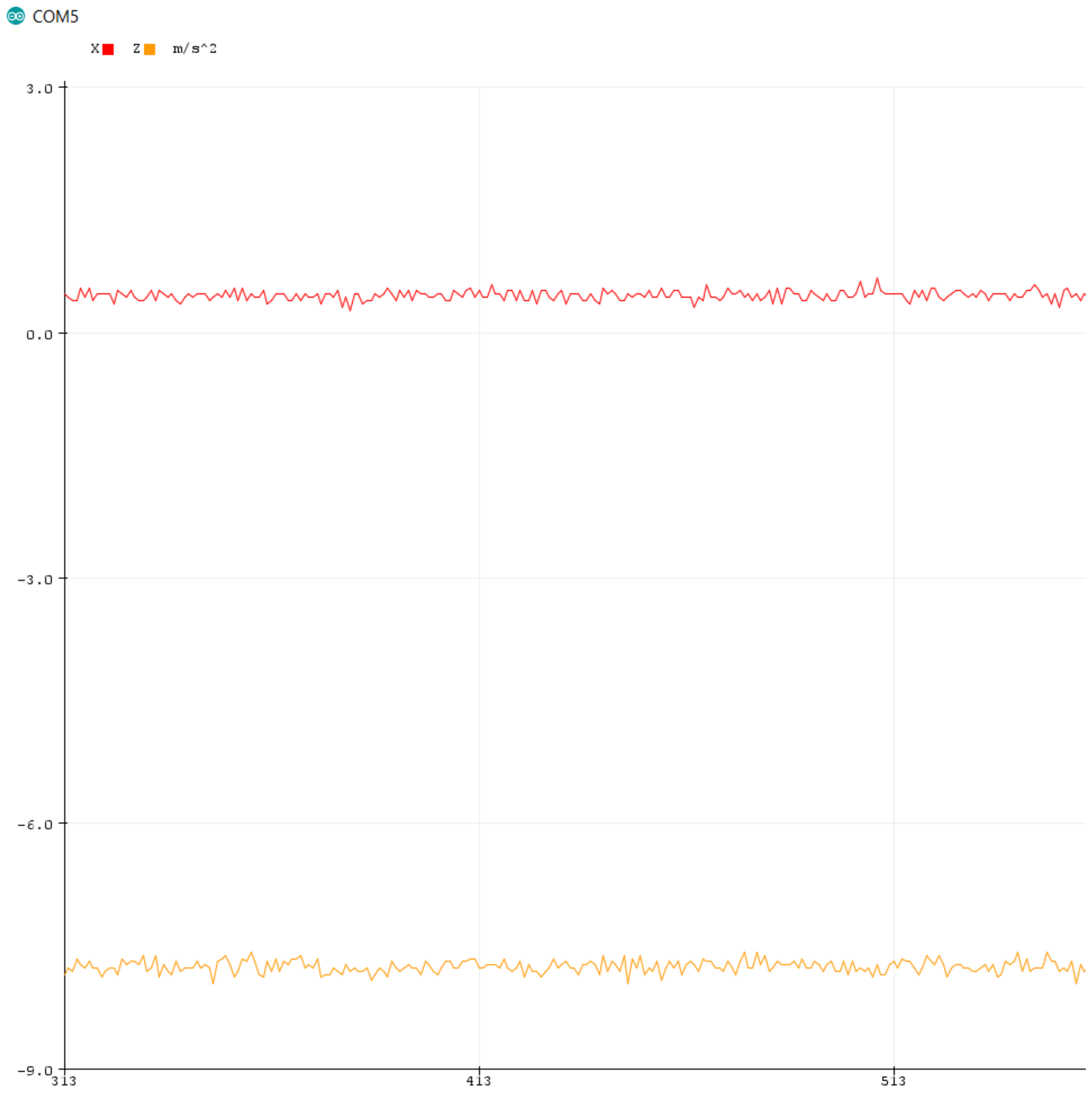


Figure 4.4. Acceleration vs time graph when there is no vibration

In a nutshell with the test and experiments we have made with the system we have seen that acceleration on the vibration plate is not reducing but frequency is increasing as we apply the force via springs on it to outwards. In this project our aim was to see what type of effects occurs during on the vibration plate. For further studies, the stiffness matrixes can be used to analyze the system with rotating each stepper motor at different rotations. We have seen the displacement of the plate is reducing when take a video shot on high frame rate as can be seen in the USB flash disk which will be also given as an appendix.

In this project we have learned how to manage an engineering project process. Also get a lot acknowledge about manufacturing process, coding and electrical components. Besides that, we experienced how we use them in an efficient way when we study on our project. We had a lot of problems during project about the mechanical and electronic design but mostly with electronic components. It was so helpful to us for learning because trying to solve problems also teach you much.

REFERENCES

[1] https://en.wikipedia.org/wiki/Laser_cutting (January, 2020)

[2] https://www.mnbprecision.com/basics-wire-erosion (January, 2020)

[3] Rose I. (2014) Nickel Plating Handbook, Nickel Institute (NI), Brussels, Belgium

[4] https://atunkimya.com/surec/black-phosphate-coating (January, 2020)

[5] https://www.gsproducts.co.uk/blog/tag/m6-m8-m10-m12-m16-m20-m24-m30-m36-m42-thread-pitch (January, 2020)

[6] https://opensource.com/resources/what-arduino (January, 2020)

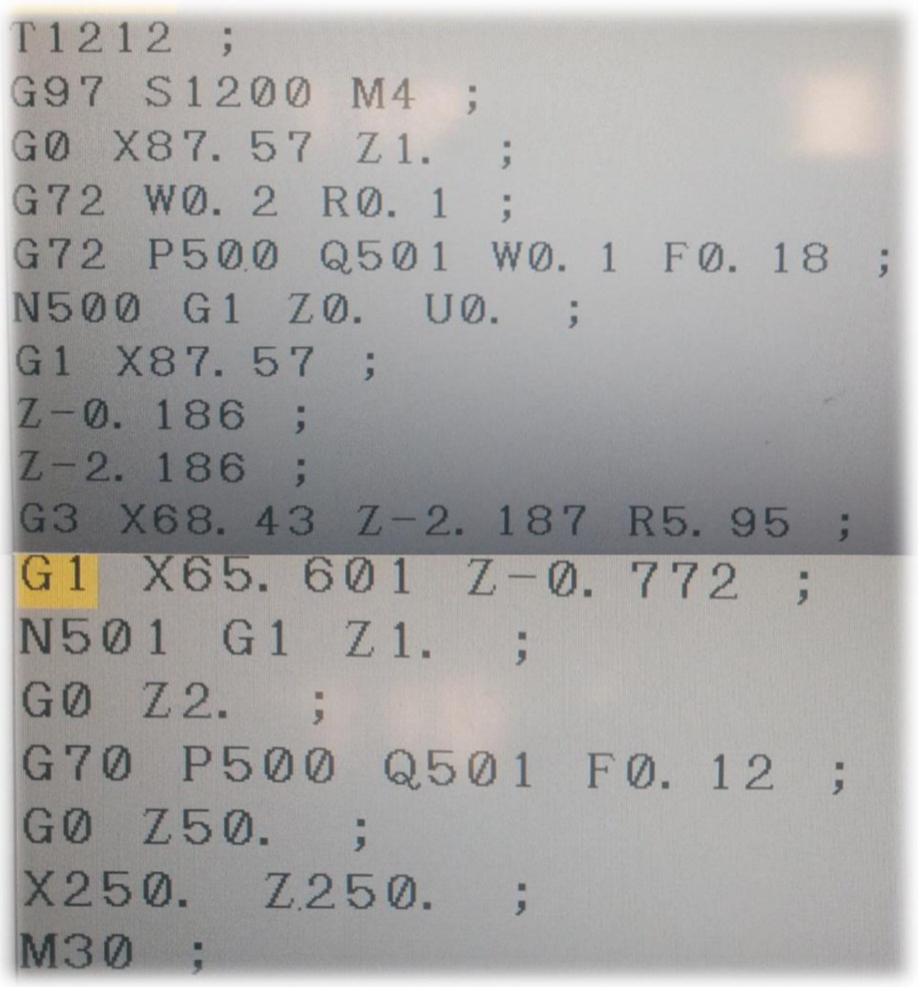
[7] Erdinç F. (March, 2016) Yeni Başlayanlar için Arduino, 2nd Edition, Pusula 20 Teknoloji ve Yayıncılık A.Ş., Ümraniye, İstanbul

[8] Delebe E. (August, 2017) Projeler ile Arduino, 11th Edition, İnkılap Kitabevi Yayın San. Tic. A.Ş., Bağcılar, İstanbul

[9] Rivin E. (1999) Stiffness and Damping in Mechanical Design, Marcel Dekker Inc., Basel, Switzerland

APPENDICES

Appendix I – CNC code for channel operation on the vibration plate



```
T1212 ;
G97 S1200 M4 ;
G0 X87.57 Z1. ;
G72 W0.2 R0.1 ;
G72 P500 Q501 W0.1 F0.18 ;
N500 G1 Z0. U0. ;
G1 X87.57 ;
Z-0.186 ;
Z-2.186 ;
G3 X68.43 Z-2.187 R5.95 ;
G1 X65.601 Z-0.772 ;
N501 G1 Z1. ;
G0 Z2. ;
G70 P500 Q501 F0.12 ;
G0 Z50. ;
X250. Z250. ;
M30 ;
```

T1212 for tool number and offset

G97 for constant revolution

M4 for counter clockwise direction

G0 for fast motion while closing the material

X57.57 for radius starting point

Z1. for safe distance

G72 for face lathe operation loop

W0.2 for pass value for each cycle

R0.1 for back run distance after each pass

P500 for loop starting code

Q501 for loop stopping code

W0.1 for finish pass value on z axis

F0.18 cutting speed

N500 for loop starting code

G1 for slow motion

Z0. for face of the material

U0. for moving through uncut material

G3 for radius movement on the counter clockwise direction

X68.43 for radius ending point

Z-2.187 for radius ending point

R5.95 for radius (ours is 6.35 but there is 0.4 radius on the edge of the cutting diamond)

X65.601 for exiting point

Z-0.772 for exiting point

N501 for loop ending code

Z1. for going to safe distance, starting point

Z2. for going to safe distance, starting point

G70 for finish lathe code

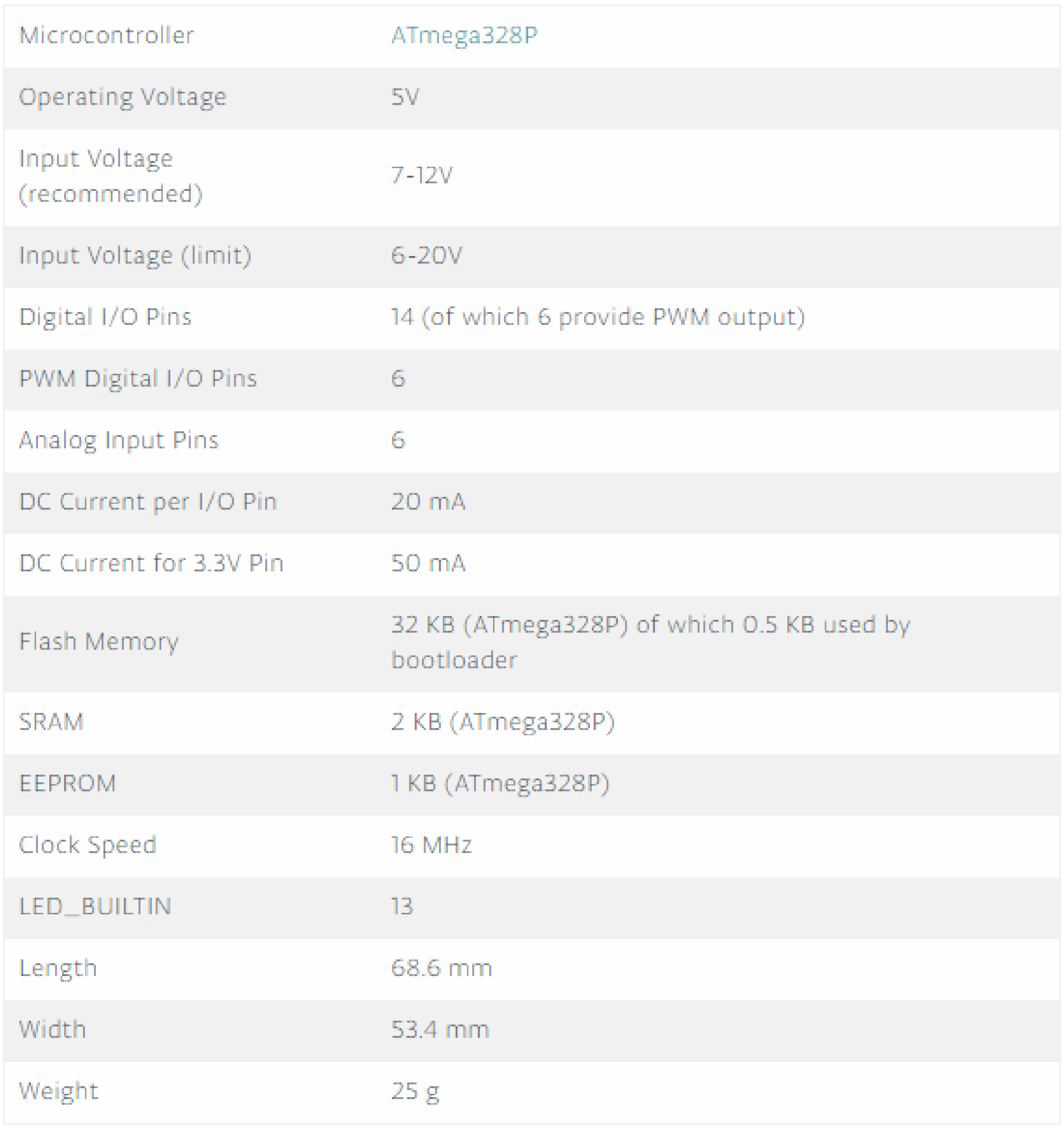
P500 Q501 for the cycle will finished

G0 Z50. for getting away from the material

X250. Z250. for safe distance to turret turning

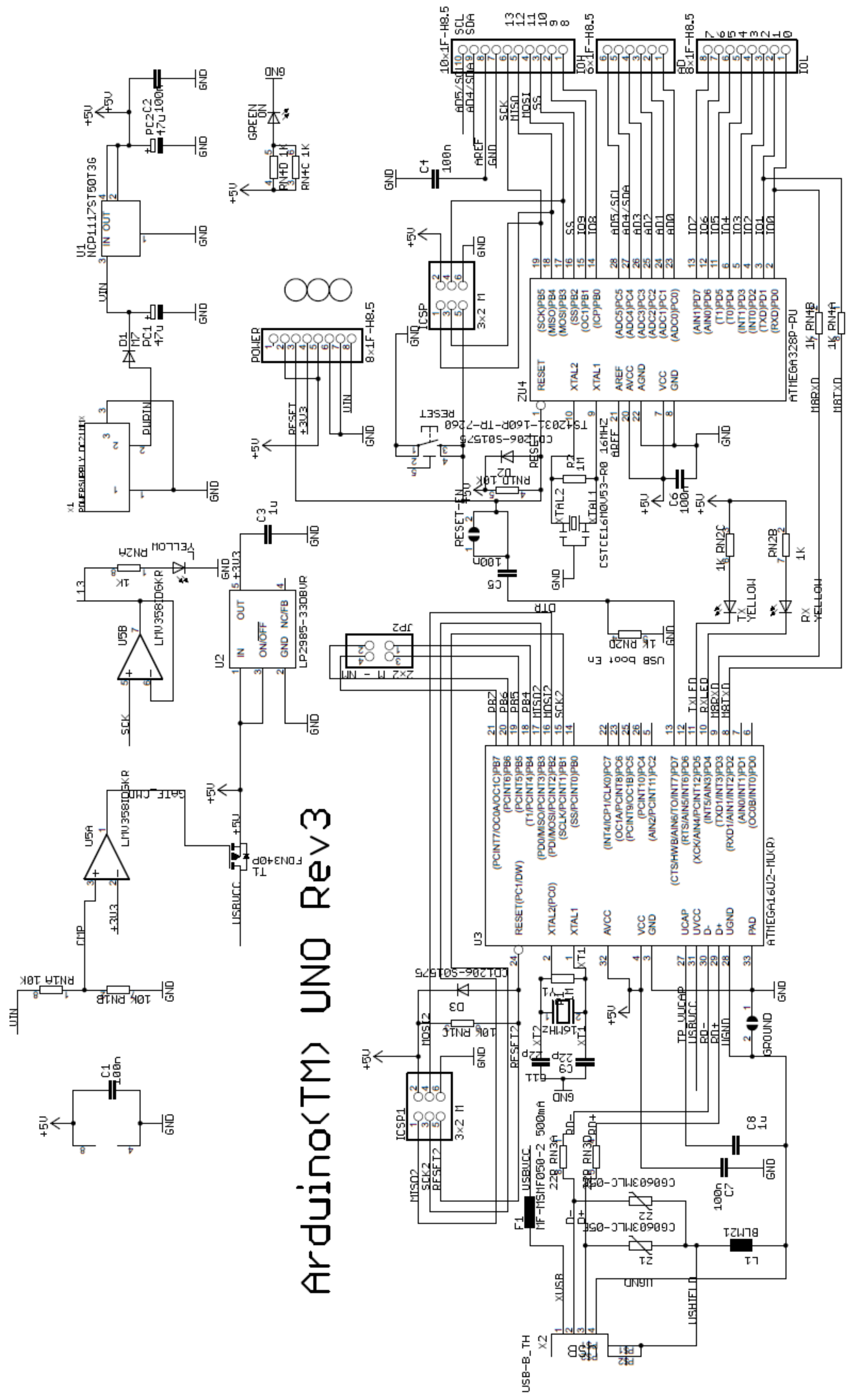
M30 for ending the program

Appendix II.A – Technical Specifications of Arduino Uno

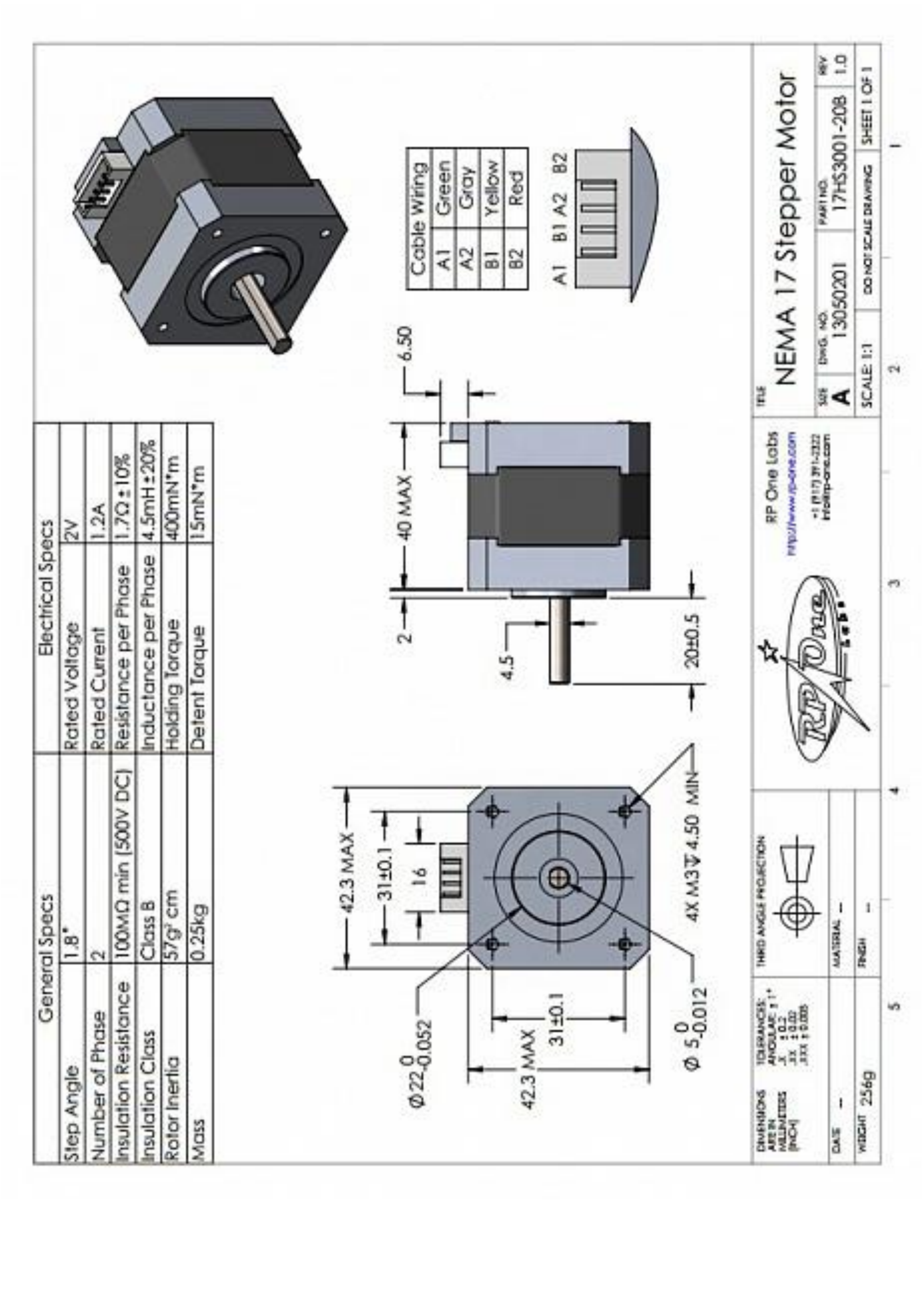


Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Appendix II.B – Schematic of Arduino Uno



Appendix III – Datasheet of Nema 17HS3001 Stepper Motor



General Specs		Electrical Specs	
Step Angle	1.8°	Rated Voltage	2V
Number of Phase	2	Rated Current	1.2A
Insulation Resistance	100MΩ min (500V DC)	Resistance per Phase	1.7Ω ± 10%
Insulation Class	Class B	Inductance per Phase	4.5mH ± 20%
Rotor Inertia	57g² cm	Holding Torque	400mN*m
Mass	0.25kg	Detent Torque	15mN*m

Cable Wiring	
A1	Green
A2	Gray
B1	Yellow
B2	Red

DIMENSIONS ARE IN MILLIMETERS (INCH)	TOLERANCES: ANGULAR: ± 1° .X ± 0.2 .XX ± 0.02 .XXX ± 0.005	THIRD ANGLE PROJECTION	RP One Labs http://www.rp-one.com +1 (917) 391-2322 info@rp-one.com
DATE —		MATERIAL —	
WEIGHT 256g		FINISH —	

TITLE			
NEMA 17 Stepper Motor			
SIZE A	DWG. NO. 13050201	PART NO. 17HS3001-20B	REV 1.0
SCALE: 1:1	DO NOT SCALE DRAWING	SHEET 1 OF 1	

Appendix IV – Datasheet of WRK-450SH Vibration DC Motor

WRK-450SH

OUTPUT:APPROX 0.5W~5.0W

Carbon-brush motors

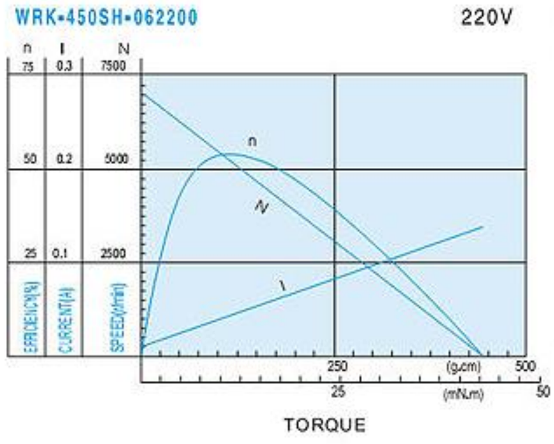
輸出功率：約0.5W~5W

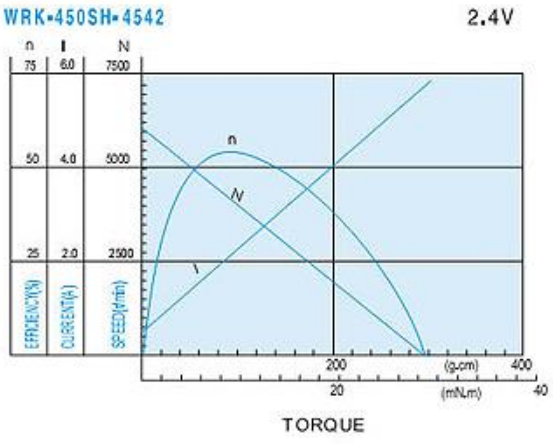
碳刷馬達

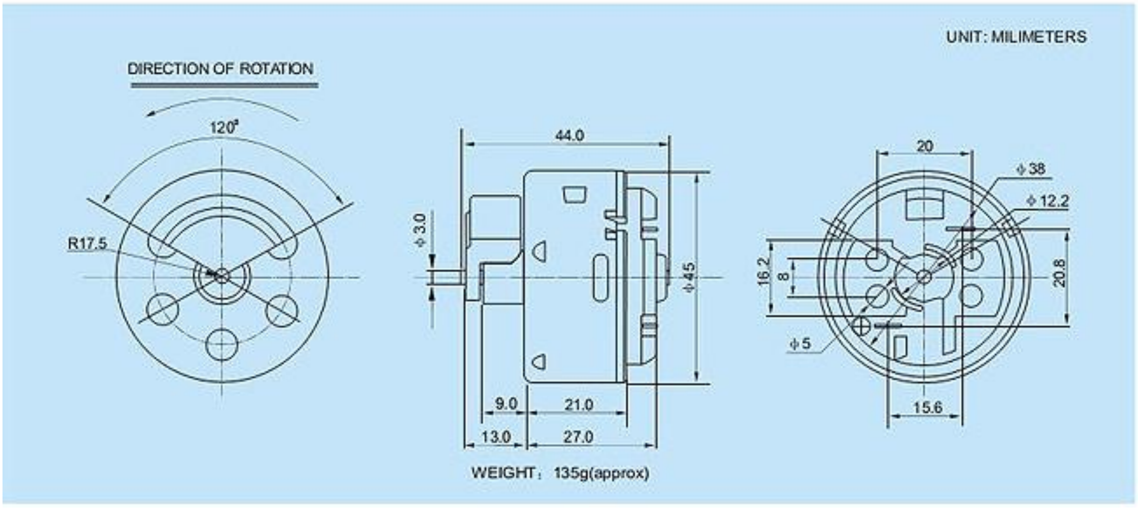
典型應用／家用電器：按摩器產品

Typical Application/ Household Appliance Massage/Vibrator

型 号 MODEL	电 压 VOLTAGE		无负荷 NO LOAD		最大效率点 AT MAXIMUM EFFICIENCY					起 动 STARTING		
	使用范围 OPERATING RANGE	额定值 NOMINAL	转速 SPEED	电流 CURRENT	转速 SPEED	电流 CURRENT	转 距 TORQUE		功率 OUTPUT	转 距 TORQUE		电流 CURRENT
			r/min	A	r/min	A	g · cm	mN · m	W	g · cm	mN · m	A
WRK-450SH-062200	200~240	220V	6900	0.010	5600	0.035	877	86	4.9	4489	440	0.14
WRK-450SH-4542	1.5~3.0	2.4V	6100	0.50	4800	1.65	551	54	2.66	2551	250	5.82







Appendix V.A – Accelerometer and Vibration DC motor code on Arduino IDE

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
int motor_Pin=5,Motor_Speed=0;
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);
void setup() {
        Serial.begin(9600);
        if(!accel.begin())
        {
        Serial.println("No ADXL345 detected, check your wiring!");
        while(1);
         }
        accel.setDataRate(ADXL345_DATARATE_200_HZ);
        accel.setRange(ADXL345_RANGE_16_G);
        pinMode(motor_Pin, OUTPUT);
        }
void loop() {
        Motor_Speed=map(analogRead(A1), 0, 1023,0, 50);
        analogWrite(motor_Pin, Motor_Speed);
        sensors_event_t event;
        accel.getEvent(&event);
        Serial.print("X: "); Serial.print(event.acceleration.x); Serial.print("  ");
        Serial.print("Y: "); Serial.print(event.acceleration.y); Serial.print("  ");
        Serial.print("Z: "); Serial.print(event.acceleration.z);
        Serial.print("  ");Serial.println("m/s^2 ");
        delay(1);
        }
```

Appendix V.B – Stepper motors, Joystick and LCD screen code on Arduino IDE

```
#define DIR1       6
#define STEP1      7
#define DIR2      10
#define STEP2     11
#define DIR3      12
#define STEP3    13
#include <LiquidCrystal_I2C_AvrI2C.h>
LiquidCrystal_I2C_AvrI2C lcd(0x27,16,2);
boolean dir_bool = LOW,setdir;
int button_dir=3,speed,position_reset=2;
int Position=0,Analog_Value=0,cycle=0,old_cycle=0;
void setup() {
        Serial.begin(9600);
        pinMode(button_dir,INPUT);
        pinMode(position_reset, INPUT_PULLUP);
        pinMode(DIR1,OUTPUT);
        pinMode(STEP1,OUTPUT);
        pinMode(DIR2,OUTPUT);
        pinMode(STEP2,OUTPUT);
        pinMode(DIR3,OUTPUT);
        pinMode(STEP3,OUTPUT);
        lcd.begin();
        lcd.clear();
        delay(5);
        lcd.setCursor(0,0);
        lcd.print("REVOLUTION 0");
        }
void reset(){
        speed=1500;
        while(Position!=0){
        if(Position>0){
        Position--;
        digitalWrite(DIR1,LOW);
        digitalWrite(DIR2,LOW);
        digitalWrite(DIR3,LOW);
        digitalWrite(STEP1,HIGH);
        digitalWrite(STEP2,HIGH);
        digitalWrite(STEP3,HIGH);
        delayMicroseconds(speed);
        digitalWrite(STEP1,LOW);
        digitalWrite(STEP2,LOW);
        digitalWrite(STEP3,LOW);
        delayMicroseconds(speed);
   }else{
         Position++;
         digitalWrite(DIR1,HIGH);
         digitalWrite(DIR2,HIGH);
         digitalWrite(DIR3,HIGH);
         digitalWrite(STEP1,HIGH);
         digitalWrite(STEP2,HIGH);
         digitalWrite(STEP3,HIGH);
         delayMicroseconds(speed);
         digitalWrite(STEP1,LOW);
         digitalWrite(STEP2,LOW);
         digitalWrite(STEP3,LOW);
         delayMicroseconds(speed);
     }
      old_cycle=Position/200;
      if(old_cycle!=cycle)
     {
      cycle = old_cycle;
      lcd.clear();
      delay(5);
```

```
            lcd.setCursor(0,0);
            lcd.print("Revolution ");
            lcd.print(cycle, DEC);
           }
     }
}
void loop() {
               if(!digitalRead(position_reset))
                reset();
                Analog_Value=analogRead(A1);
                if(Analog_Value>562){
               speed=map(Analog_Value, 562, 1023,4000, 1000);
 Position++;
   digitalWrite(DIR1,HIGH);
   digitalWrite(DIR2,HIGH);
   digitalWrite(DIR3,HIGH);
   digitalWrite(STEP1,HIGH);
   digitalWrite(STEP2,HIGH);
   digitalWrite(STEP3,HIGH);
   delayMicroseconds(speed);
   digitalWrite(STEP1,LOW);
   digitalWrite(STEP2,LOW);
   digitalWrite(STEP3,LOW);
   delayMicroseconds(speed);
  }
  else if(Analog_Value<462){
   speed=map(Analog_Value, 0, 462,1000, 4000);
Position--;
   digitalWrite(DIR1,LOW);
   digitalWrite(DIR2,LOW);
   digitalWrite(DIR3,LOW);
   digitalWrite(STEP1,HIGH);
```

```
    digitalWrite(STEP2,HIGH);
    digitalWrite(STEP3,HIGH);
    delayMicroseconds(speed);
    digitalWrite(STEP1,LOW);
    digitalWrite(STEP2,LOW);
    digitalWrite(STEP3,LOW);
    delayMicroseconds(speed);
  }
  if(Position<0){
          old_cycle=Position/200;
          if(old_cycle!=cycle){
          cycle = old_cycle;
          lcd.clear();
          delay(5);
          lcd.setCursor(0,0);
          lcd.print("REVOLUTION ");
          lcd.print(-cycle, DEC);
          }
  }else{
          old_cycle=Position/200;
             if(old_cycle!=cycle){
            cycle =(old_cycle);
            lcd.clear();
            delay(5);
            lcd.setCursor(0,0);
            lcd.print("REVOLUTION ");
            lcd.print(-cycle, DEC);
       }
    }
}
```