MARMARA UNIVERSITY
FACULTY OF ENGINEERING

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

Cebrail TOPAL
Ali Alperen ERKOÇ

GRADUATION PROJECT REPORT
Department of Mechanical Engineering

Supervisor

Dr. Ömer Haluk BAYRAKTAR

ISTANBUL, 2022

MARMARA UNIVERSITY
FACULTY OF ENGINEERING

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

by

Cebrail TOPAL - Ali Alperen ERKOÇ

(150416043) (150416049)

June, 2022, 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, we would like to thank my supervisor Dr. Ömer Haluk BAYRAKTAR, for the valuable guidance and advice on preparing this thesis and for giving me moral and material support.

Besides, Dr. Ömer Haluk BAYRAKTAR we would like to thank, as well all the professors and research assistants for their training, problem-determination, solving, and analyzing skills as an engineer and their experiences.

June 2022

Cebrail TOPAL
Alperen ERKOÇ

TABLE OF CONTENTS

SYMBOLS

a: Acceleration

F: Force

g: Acceleration of gravity

k: Stiffness

mm: Millimeter

ms: Millisecond

t: Thickness

V: Volt

W: Watt

ABBREVIATIONS

A: Ampere

DC: Direct Current

D.O.F.: Degree of Freedom

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

LIST OF TABLES

1. INTRODUCTION

Throughout human history, studies have been carried out on structures that will distribute heavy loads to various points and be robust. Especially in the construction sector, this issue has been given great importance and cage systems have been developed. In addition to its robustness, the cage system elements are also used in robotic systems by adding mobility. For example, these elements can be used as parallel manipulators when designing a robot arm.

First, we should explain parallel manipulators. A parallel manipulator is a computer-controlled mechanical system that uses different series of chains to support a single platform or end tool. The best-known parallel manipulators are systems that support a base that can act as a flight simulator with six linear actuators. Also known as parallel robots, these systems are articulated robots that use similar mechanisms to move the underlying robot or one or more manipulators. The word parallel in the name is because, unlike serial manipulators, the last tool is supported by more than one connection. The word parallel is not used in a geometric sense. Parallel does not mean that the connection lines are parallel to each other, but that the connections move together.

Most robot applications need rigidity. Serial robots provide this rigidity with high-quality rotary joints that can move on one axis but maintain their stiffness in other movements. Every joint that allows movement must do this movement under the planned control of an actuator. Therefore, a movement that requires several axes needs that many joints. Undesirable flexibility or incline in a joint causes the same inclination and flexibility to be experienced in the arm. It is not possible for one joint to compensate for the movement of the other.

1.1. Literature Survey

Kinematics of Parallel Manipulators

Many researches over parallel manipulators have been carried out over the last few decades.

Inoue et al. **[1]** proposed and investigated an octahedral parallel manipulator, manipulated by rotary actuators instead of linear, providing much wider range of workspace and much better back-drivability. Their work included the mechanical design, kinematical analysis, static force analysis, control system implementation and compliance in hand coordinate system of the mechanism.

Wang and Wang **[2]** demonstrated a different design of the platform and have completed an inverse kinematics analysis with acceptable tolerances, using an approximate distribution model for the position and orientation.

Naccaratto **[3]** has contributed a complete analysis on Kinematics of VGTM's for his PhD. His work included a closed-form solution using an expanded version of the link length method.

Dong, Du and Sun **[4]** have designed and experimented a novel dual macro/micro wide-range flexure hinge-based parallel mechanism. They have succeeded a repeatability and resolution to nanometer scale.

Dynamics of Parallel Manipulators

Cho, Tesar and Freeman **[5]** developed a general dynamic modeling technique for general robotic manipulator systems with antagonistic actuation, using kinematical influence coefficients. Authors extended their work to include a stiffness characteristics model to accomplish robust operation under disturbances. The proposed system is a general kinematic system with hybrid serial/parallel type actuations.

Rong and Zong Guanghua **[6]** have contributed an in depth investigation on the dynamics of parallel mechanism, based on Influence Coefficient (KCI) Method.

Stiffness of Mechanisms

Alici and Shirinzadeh **[7]** presented an enhanced stiffness modelling, identification and characterization for general manipulators, also counterprooving, by accurate experimentation, the conventional stiffness modelling is incomplete. The new modelling includes the effects of manipulator configuration and loading on stiffness. It has been shown that although the component of the stiffness matrix differentiating the enhanced stiffness model from the conventional one is not always positive definite, the resulting stiffness matrix can still be positive definite.

Late studies over stiffness representation of mechanisms have prospered, yielding newby mathematical applications on the synthesis of more efficient models. Upon this aim, recent tools have been developed and used. One of the most promising is the screw theory. *Robert Stawell Ball* **[8]** first introduced screws for the analysis of ridig body motion. *Dimentberg* **[9]** first used screw theory in constructing the stiffness matrix of spring systems in unloaded equilibrium. *Loncaric* **[10]** used Lie algebra to analyze stiffness and compliance.

Roberts **[11]** has shown that any symmetric positive semi-definite matrix can be written in Loncaric's normal form and be represented by a minimal parallel connection of simple and screw springs with concurrent axes intersecting at a center of stiffness. As an application this result has been used to design a compact parallel compliance mechanism with a prescribed positive semi-definite spatial stiffness matrix.

2. METHOD

2.1. Analytic Calculations and Modelling

In this study, we do a stiffness manipulation of 2D parallel planar manipulators having 3 linear actuations. We do this manipulation at any point at the workspace of the end-effector.

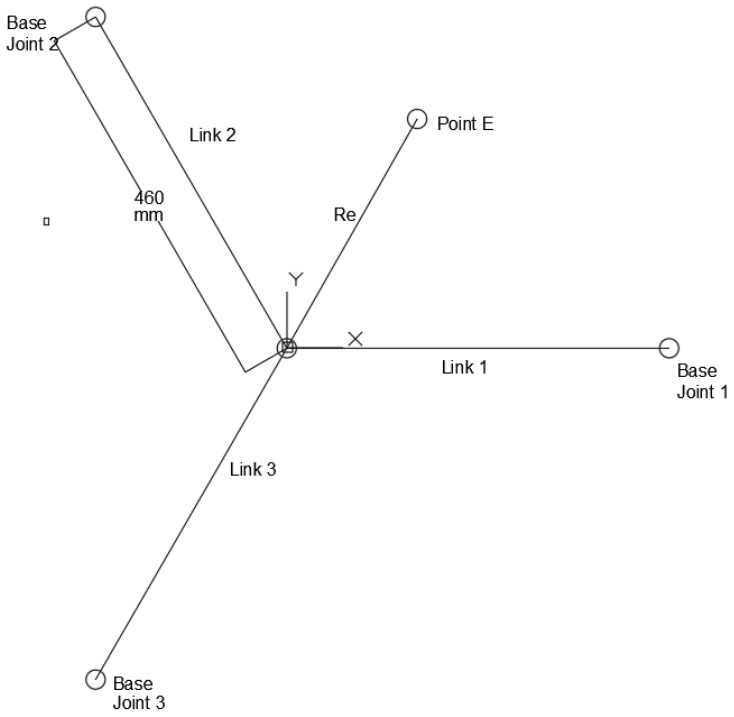


Figure 2.1 Link positions when the system at origin.

$\overline{L}$ is a vector defining the lengths of actuated links of the mechanism and $\bar{r}$ is the position vector of base joints.

$$\overline{L} = \begin{bmatrix} L_1 & L_2 & L_3 \end{bmatrix}^T \tag{2.1}$$

$$\overline{r} = \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}^T \tag{2.2}$$

$\overline{r_E}$ is the position vector of the point P at point E. For the case of this planar manipulators with 2dof, $\overline{r_E}$ vector will be represented as:

$$\overline{r_E} = \begin{bmatrix} r_x \, r_y \end{bmatrix}^T \tag{2.3}$$

Each L distance is a function of $\overline{r_E}$. Therefore, derivation of $\overline{L}$ vectors with respect to $\overline{r_E}$ gives us a matrix called Jacobian. Which is numerically specific for any point E. Which is numerically specific for ant point E. Note: Analytically, it represents a certain mechanism.

$$L_1 = \sqrt{(x_1 - x_E)^2 + (y_1 - y_E)^2} \tag{2.4}$$
$$L_2 = \sqrt{(x_2 - x_E)^2 + (y_2 - y_E)^2}$$
$$L_2 = \sqrt{(x_3 - x_E)^2 + (y_3 - y_E)^2}$$

For the connection between $\overline{L}$ and $\overline{r_E}$ vectors we use the following relation (Jacobian).

$$\mathcal{J} = \frac{d\overline{L}}{d\overleftarrow{r_E}} \tag{2.5}$$

$$d\overline{L} = \mathcal{J} * d\overleftarrow{r_E} \tag{2.6}$$

From Eq. 2.5, Jacobian for our specific system is constructed as follows:

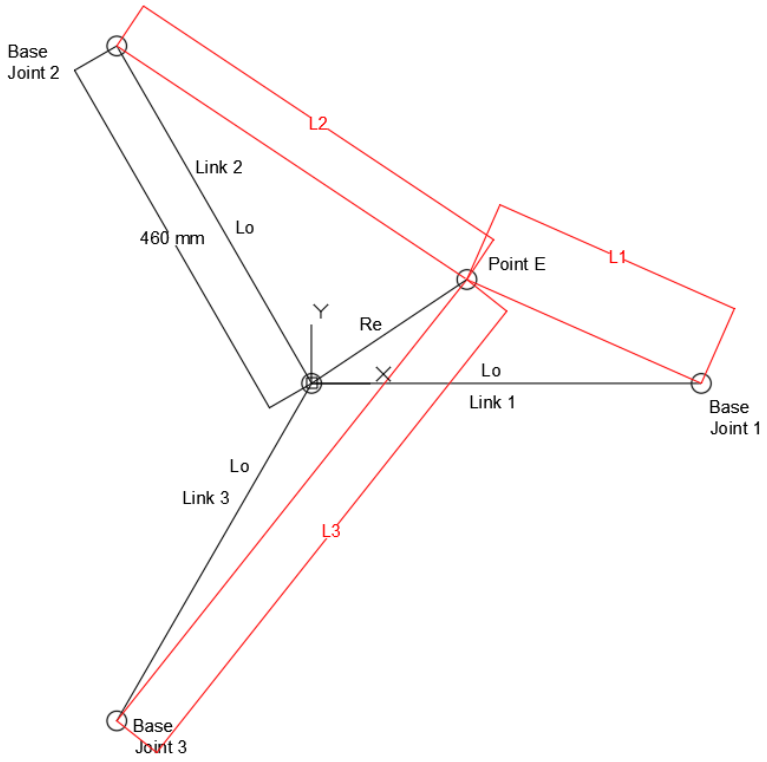


Figure 2.2 Link lengths when the end-effector at point E

Length of each link is 460 mm when the end-effector is at point P (origin). The angle between each link is 120^o. By using this geometry, we can calculate x and y components of joints.

$$\begin{aligned} L_0 &= 460mm \\ x_1 &= L_0 \\ x_2 &= L_0 * \cos 120 \\ x_3 &= L_0 * \cos 240 \\ y_1 &= L_0 * \sin 180 \\ y_2 &= L_0 * \sin 120 \\ y_3 &= L_0 * \sin 240 \end{aligned} \tag{2.7}$$

To calculate L_f, we use the Eq. 2.4 to calculate the link lengths at E point.
x_1, x_2, x_3, y_1, y_2 and y_3 values calculated on Eq. 2.7.

From now we have $\overline{L}$ vectors which are the functions of $\overline{r_E}$. If we take derivative of $\overline{L}$ with respect to $\overline{r_E}$, we will get the Jacobian matrix.
Using Eq. 2.5 and 2.6 we get:

$$J = \begin{bmatrix} \frac{dL_1}{dx_E} & \frac{dL_1}{dy_E} \\ \frac{dL_2}{dx_E} & \frac{dL_2}{dy_E} \\ \frac{dL_3}{dx_E} & \frac{dL_3}{dy_E} \end{bmatrix} \tag{2.8}$$

After all these equations, now we can calculate the Jacobian matrix for any E point we want to move the end-effector from eq. 2.9.

$$J = \begin{vmatrix} \frac{x_E - 46}{\sqrt{x_E^2 - 92x_E + y_E^2 + 2116}} & \frac{y_E}{\sqrt{x_E^2 - 92x_E + y_E^2 + 2116}} \\ \frac{x_E + 23}{\sqrt{(x_E + 23)^2 + (y_E - 23\sqrt{3})^2}} & \frac{y_E - 23\sqrt{3}}{\sqrt{(x_E + 23)^2 + (y_E - 23\sqrt{3})^2}} \\ \frac{x_E + 23}{\sqrt{(x_E + 23)^2 + (y_E + 23\sqrt{3})^2}} & \frac{y_E + 23\sqrt{3}}{\sqrt{(x_E + 23)^2 + (y_E + 23\sqrt{3})^2}} \end{vmatrix} \tag{2.9}$$

If we want to move the end-effector from to any E point, we need to calculate what link length difference at E point for each link will be.
We are using the $\overline{L}$ vectors to find link length difference. Initial link length matrix is like:

$$\overline{L}_i = \begin{bmatrix} L_0 \\ L_0 \\ L_0 \end{bmatrix} \tag{2.10}$$

We have the initial link length matrix, and we have the $\overline{L}$ matrix which defines the link lengths when the end-effector at any E point. Using the following relation to achieve the link length difference at E point.

$$dL = L - L_i \tag{2.11}$$

With this calculation we know how much we need to change the link lengths to move the end-effector to point E.
But our system runs with stepper motors. Because of that we need to convert dL values to step values. When our stepper motors run 200 steps, our links stretch out 1 mm. From that

information, by multiplying dL value with 200 we calculate the number of steps that the stepper motors should run.

$$S = dL * 200 \tag{2.12}$$

All these calculations help us to move end-effector to any point we want. After this part we will be able to manipulate stiffness of the end-effector at any point E.
Representing force acting on the end-effector as,

$$\overline{F} = \begin{bmatrix} F_x \\ F_y \end{bmatrix} \tag{2.13}$$

Virtual work on system can be written as follows.

$$dW = F_x d_x + F_y d_y \tag{2.14}$$

Where d_x and d_y are the difference on position vectors. They can be written in a matrix named fr.

$$\overline{fr} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} \tag{2.15}$$

So, the virtual work matrix becomes:

$$fW = \overline{F}^T fr \tag{2.16}$$

We can write the fW function by using link reaction forces. Where fL represents the extension of the springs and $\overline{T}$ is the link reaction matrix.

$$fW = T^T fL \tag{2.17}$$

By combining equations 2.16 and 2.17, we get the relation as follows.

$$\overline{F^T}\,\overline{fr} = \overline{T^T}\,\overline{fL} \tag{2.18}$$

$$F^T = T^T \frac{fL}{fr} \tag{2.19}$$

We know that the expression $\frac{fL}{fr}$ gives the Jacobian. So, the equation becomes:

$$\overline{F} = \mathcal{J}^T \overline{T} \tag{2.20}$$

We can also write $\overline{F}_{ext}$ and $\overline{T}$ in a different way.

$$T = K_L fL \tag{2.21}$$

Which corresponds to potential energy.

$$u = \frac{1}{2} ks^2 = K_L fL \tag{2.22}$$

$$F = K_E fr \tag{2.23}$$

Where K_E is resultant stiffness at the end-effector. So, if we control K_L, this means we can manipulate the stiffness at the end-effector.
For example, if we increase K_L for the constant T, fL and fr will decrease but K_E will increase. This means our manipulation on K_L will affect K_E proportionally.

If we go back to equation 2.21, multiple sets of $\overline{T}$ may give the same $\overline{F}$ outputs.
The difference of these multiple $\overline{T}$ sets is that it increases the stiffness of the system, while obeying 2.21 thought $\mathcal{J}$. So, we can add $T_{ext} + T_{\text{int}}$ to get $\overline{F}$ still.

$$T = T_{ext} + T_{\text{int}} \tag{2.24}$$

By combining Eq. 2.20 and 2.24 we get the following relations.

$$J^{-T} F = T_{ext} \tag{2.25}$$

$$\varnothing = J^T T_{\text{int}} \tag{2.26}$$

When Eq. 2.26 is simplified, we get,

$$T_{\text{int}} = \begin{bmatrix} \tilde{n} \\ \tilde{I} \end{bmatrix} \bar{\lambda} \tag{2.27}$$

Here, $\bar{\lambda}$ is any scalar vector, $\tilde{I}$ is an identity matrix and $\begin{bmatrix} \tilde{n} \\ \tilde{I} \end{bmatrix}$ is the null space matrix for J^T.

In our case, $\bar{\lambda}$ is 1x1 scalar matrix, $\tilde{I}$ =1 and $\tilde{n} \in R^{(2x1)}$. So, the Eq. 2.26 becomes,

$$J^T T_{\text{int}} = J^T \begin{bmatrix} \tilde{n} \\ \tilde{I} \end{bmatrix} \bar{\lambda} \tag{2.28}$$

In our stiffness manipulation, we use Eq. 2.21 and 2.27 to find internal T and the ΔL values. By using this ΔL value ($T = K_L \Delta L$), we are calculating the step link length difference values that our links should change to ensure the internal force. Also, we chose the $\bar{\lambda}$ matrix depending on to decide how much manipulation we will do. If we increase the $\bar{\lambda}$, our T will be increase. Depending on this, ΔL will be higher. If the ΔL is higher, this means our links will be more stretched. And that leads the higher stiffness manipulation.
As we look at the Eq. 2.28, we already constructed Jacobian matrix. We select $\bar{\lambda}$ as we want and $\tilde{I}$ is 1. In this case we need to calculate $\tilde{n}$.

In our specific system geometry, we calculate the $\tilde{n}$ as follows.

$$\tilde{n} = \begin{bmatrix} No_1 \\ No_2 \end{bmatrix} \tag{2.29}$$

$$No_1 = -\frac{\sqrt{(2x_E + 46) * (3x_E^2 - 276x_E + 3y_E^2 + 6348)}}{\sqrt{(3y_E + \sqrt{3}x_E - 46\sqrt{3}) * (46x_E + y_E^2 + 46\sqrt{3}y_E + 2116)}} \tag{2.30}$$

$$No_2 = -\frac{\sqrt{(3y_E - \sqrt{3}x_E + 46\sqrt{3}) * (x_E^2 + 46x_E + y_E^2 - 46\sqrt{3}y_E + 2116}}{\sqrt{(3y_E + \sqrt{3}x_E - 46\sqrt{3}) * (46x_E + y_E^2 + 46\sqrt{3}y_E + 2116)}} \tag{2.31}$$

While $\tilde{I}$ is 1. Our null space matrix becomes,

$$No = \begin{bmatrix} -\dfrac{\sqrt{(2x_E+46)*(3x_E^2-276x_E+3y_E^2+6348)}}{\sqrt{(3y_E+\sqrt{3}x_E-46\sqrt{3})*(46x_E+y_E^2+46\sqrt{3}y_E+2116)}} \\ -\dfrac{\sqrt{(3y_E-\sqrt{3}x_E+46\sqrt{3})*(x_E^2+46x_E+y_E^2-46\sqrt{3}y_E+2116}}{\sqrt{(3y_E+\sqrt{3}x_E-46\sqrt{3})*(46x_E+y_E^2+46\sqrt{3}y_E+2116)}} \\ 1 \end{bmatrix} \quad (2.32)$$

From Eq. 2.32 we can calculate our null space matrix for any point E.
Multiplying this null space matrix with $\overline{\lambda}$ gives us the T_{int}.
By using the relation $T = K_L \Delta L$, we calculate the ΔL where K_L is the spring coefficient.

We can convert ΔL values by multiplying 200 to step values that stepper motors should run. As Eq. 2.12 we use the following relation.

$$\overline{DeltaS} = \overline{\Delta L} * 200 \quad (2.33)$$

We can start our modeling with how to use parallel manipulators. Let's start with the logic used in the Steward platform. The moving part is given a degree of freedom with linear links. Stiffness control is possible with synchronized movement of all links. Let's start with the basics.

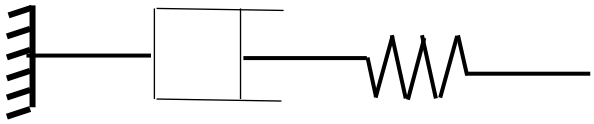


Figure 1 Modelling of The Linear Actuator

we can control position in only 1 axis with 1 linear actuator. When we add 1 linear actuator, they will create the balance of force thanks to their internal forces, but it is not possible to talk about stiffness.

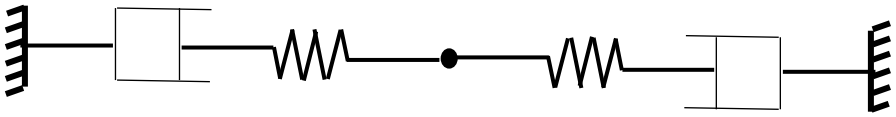


Figure 2 Modelling of The Two Linear Actuators for One Point

When we apply external force to linear actuators in equilibrium, the resistance they show will give us stiffness. In case of an external force, the balance of force will be formed again, and we will have a chance to examine the stiffness. The figure below is also shown.

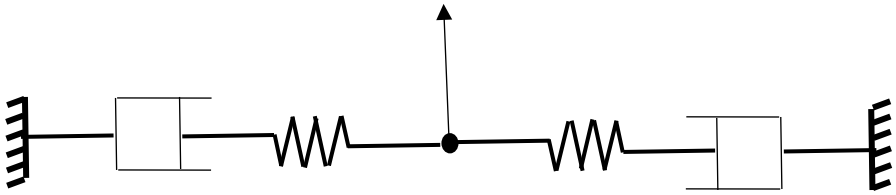


Figure 3 Modelling of The Two Linear Actuators for One Point with External Force

For this reason, when we put our linear actuators in different directions and apply external force, the internal forces will be distributed on the linear actuators at certain rates.

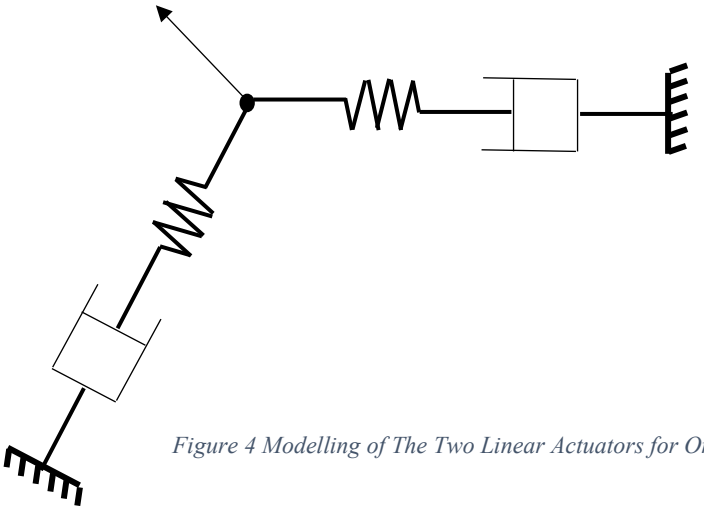


Figure 4 Modelling of The Two Linear Actuators for One Point with External Force

When we add the 3rd linear actuator, D.O.F. It will be 3. so, we will have a chance to scale our stiffness in the direction of k_{xx} or k_{yy}. The figure below is also shown.

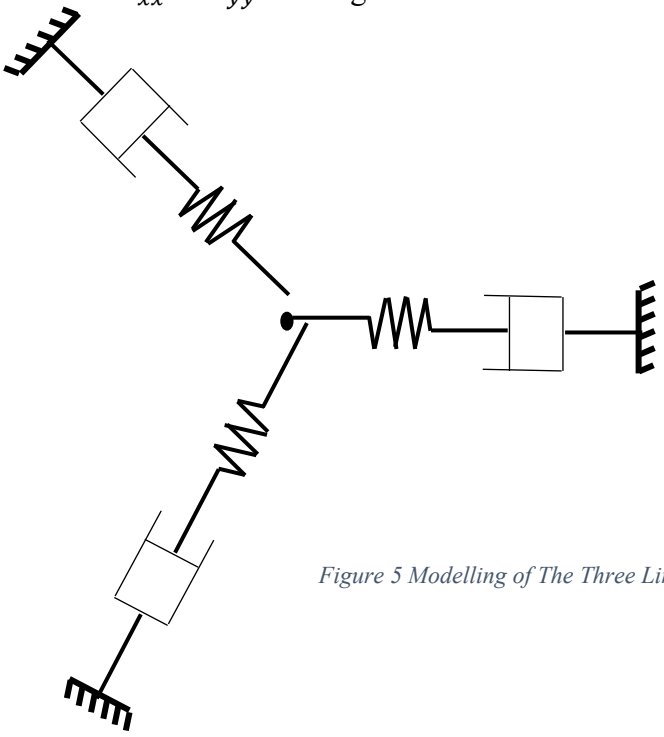


Figure 5 Modelling of The Three Linear Actuators for One Point

We mentioned that stiffness is the resistance to movement. When we look at the stiffness, we are faced with an elliptical graph. We can briefly interpret the graphs. Its axes consist of $k_{xx}$and k_{yy}. D.O.F. We can scale the stiffness because there is more than 1. If we increase it in the k_{xx} direction, the range of motion in the x direction will be more restricted. Likewise, if we increase it in the k_{yy} direction, the range of motion in the y direction will be more limited.

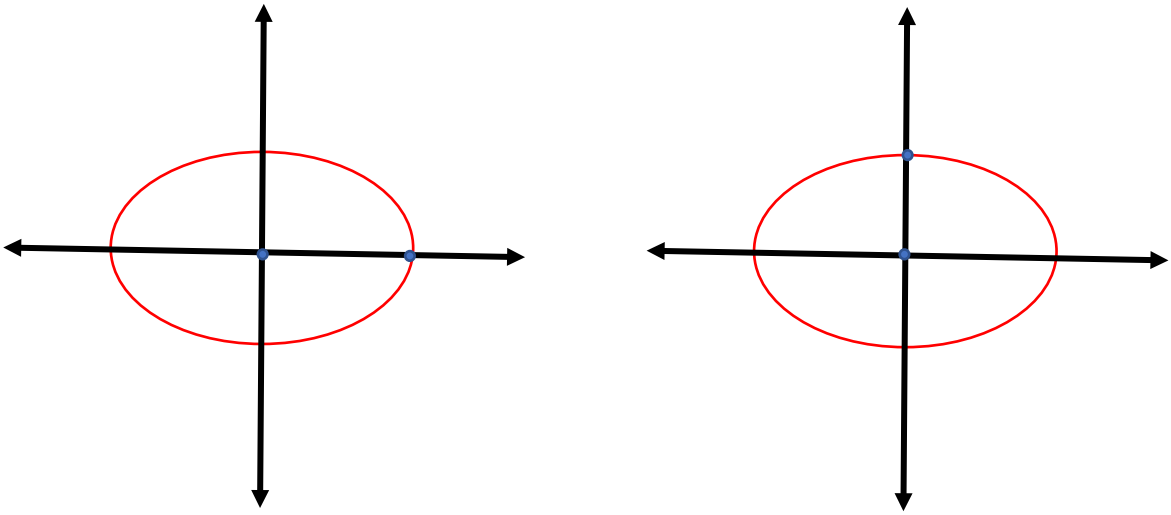


Figure 6 The Scale of Stiffness on The Graph

2.2.1. Mechanical Components

In this section, drawing and assembling the mechanical system using SolidWorks is discussed. It is very important to simulate and analyze mechanical systems before they are started. Simulating in CAD programs significantly reduces the cost of the mechanical system. In case of any problems, first improvements should be made here.

Table 1 List of Mechanical Components

#	Part Name	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 Bears	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

2.2.1.1. Main Plate

The main plate is the part on which the mechanical system is installed. The diameter has been kept wide (1000 mm) for the assembly of the whole assembly here. The choice of density is very important for the portability of the mechanical system. The material chosen should be less in mass and be resistant to any kind of movement. Aluminum was chosen for these criteria and its density was 2.73 g / cm ^ 3. If steel had been chosen, its density would be 7.86 g / cm ^ 3. We have reduced weight by 65.27% by choosing aluminum. The dimensions of the main plate are in Appendixes.

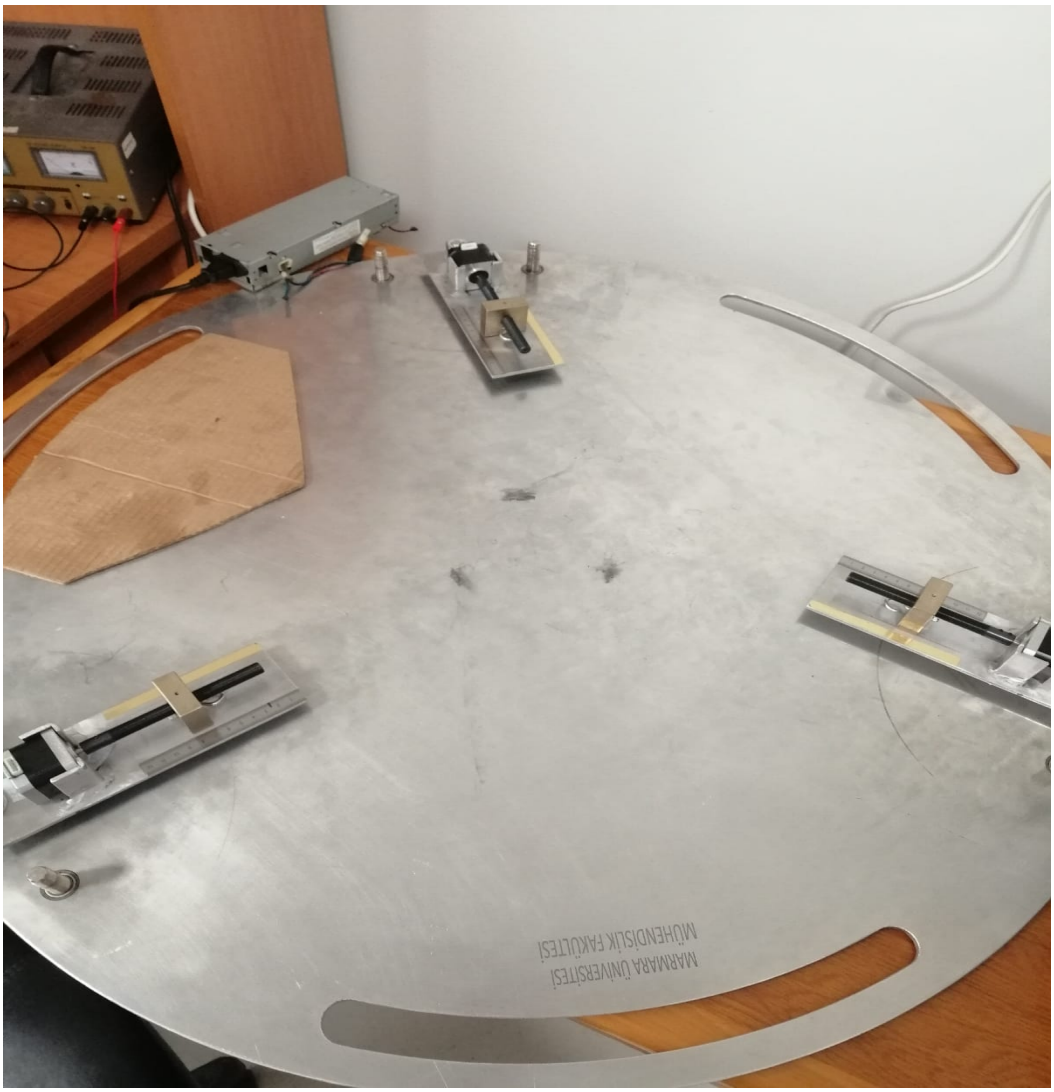


Figure 7 Main Plate

2.2.1.2. Carrier Plate

It is the parts where carrier plate step motors are assembled to the mechanical system. The carrier plate is used 3 in the mechanical system, but the number can be increased, and it provides movement in the mechanical system. Step motor is placed on this part. Then, thinned M12 shaft was assembled on the step motor. M12x1 square nut is assembled near the other end of the plate. The square nut will move over the carrier plate. As a result of this movement, a material that is difficult to wear should be selected and it is very important that it is light. As selected in the main plate, the carrier plate has been chosen as aluminum.

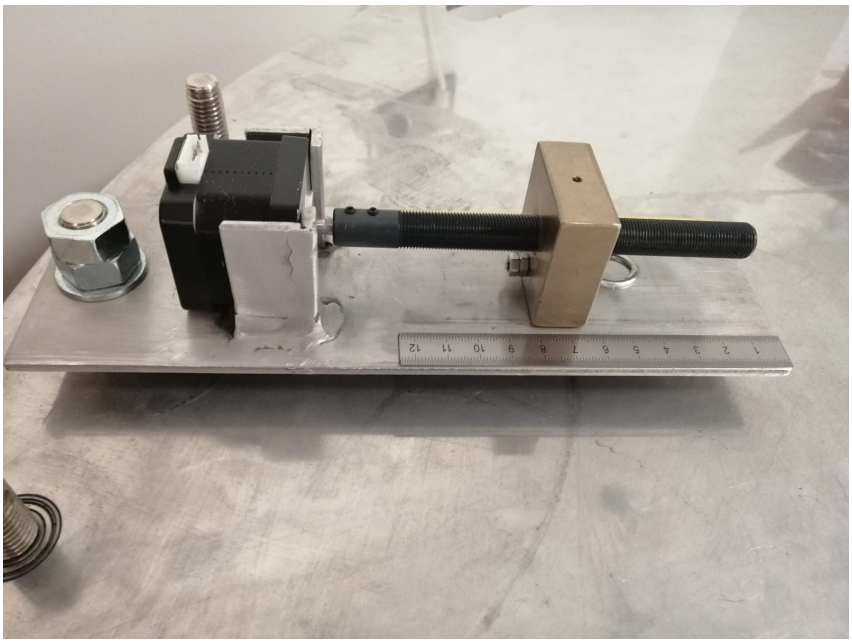


Figure 8 Carrier Plate

2.2.1.3. Vibration Plate and Clamps

The vibration plate part is the part where the DC motor and accelerometer are placed in the mechanical system. The vibration to be isolated in the system originates from this. Aluminum was chosen as the material for the vibration plate due to its durability and usefulness. The vibration plate is designed to cover the Dc motor. There is 1 clamp for each carrier plate. The clamps are likewise chosen aluminum because of their durability and lightness. The clamps can move freely in the vibration plate. Therefore, while connecting the links, it can be connected from any angle and provides ease of movement. The technical drawings of both the vibration plate and the clamps in the Appendixes.

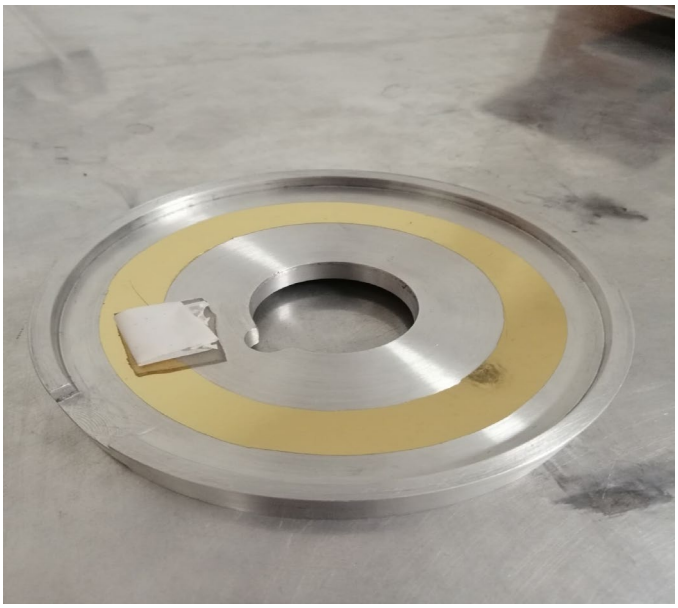


Figure 9 Vibration Plate

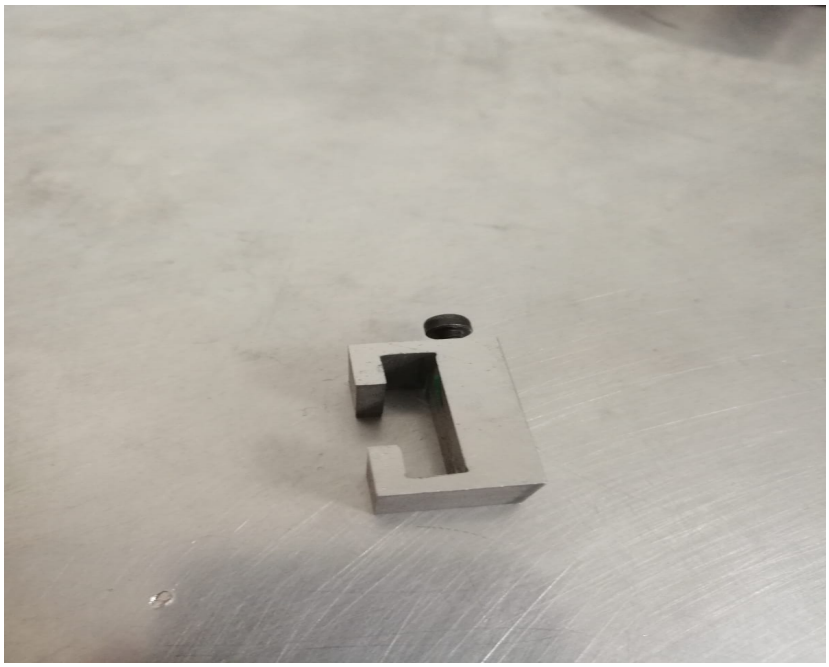


Figure 10 Clamp

2.2.1.4. M14 Pivot Bolt

M14 pivot bolt carrier plate is used while fixing it to the main plate. Steel was used in the construction of M14 pivot bolts during the manufacturing phase. The problem is that the steel may rust. As a solution to this, nickel electroplating technique was used, and this was prevented.

2.2.1.5. M12x1 Shaft and Square Nut

M12x1 shaft and square nut are very important parts. The reason of that, they are used as the connection between the vibration plate and stepper motors on the main plate. In a way, it is one of the parts that provide movement, and its importance will become more evident in the later parts of the project. 1020 steel is used as the material for the M12x1 shaft. Durability of the piece is the priority. Because the movement is based on this piece. In addition, black phosphate coating process has been applied to protect against rusting. Aluminum bronze was chosen for the M12x1 square nut. it is more useful than other materials in terms of weight.

2.2.1.6. Spring

Springs are elements that show a large elastic deformation to a degree under a certain force. If the force is removed, the element is partially or completely restored. During the deformation under load, the springs accumulate energy (deformation energy, heat) and partially give this energy back in the discharge. High strength chrome silicon silicon-manganese and chromium-vanadium alloy steels are used for springs. Also, in special cases, phosphor bronze, brass, beryllium copper and various nickel alloys can be used. Especially where great damping is required, materials such as rubber, cork, various liquids and air are used. Tension springs were used in the project and 1.4568 spring steel was chosen as the material. In material selection, it is considered that it is heat resistant and not deformed in terms of applied force.

2.2.2. Electronic Components

Another important part of the 17ort he is its software. The most convenient way to run DC motors and stepper motors is to use Arduino. Arduino is a very useful program in terms of software and hardware. The libraries used while writing the code can be accessed by anyone and this feature is very useful. C and C ++ are used as software languages. Arduino has a simpler and more understandable algorithm than other software. This is a very important reason for using Arduino. The working principle of the system is quite simple. The code is written on Arduino. There is an Arduino Uno board connected to the computer. This port is used port he codes to run. The necessary signals are transferred from the pins on the Arduino UNO and the system runs. This section provides information about electronic parts.

Many electronic parts are used in the mechanical system. detailed information is given separately. The table below lists all electronic components.

Table 2 List of Electronic Components

#	Item	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	MOSFET	14A400W PWM	1
7	12 V Power Supply	WPS-12V-16A-200W	1

2.2.2.1. Arduino Uno

Arduino uno is the most known and most used model. Anyone can do hobby, educational or professional projects with basic knowledge without having much detailed programming and electronic knowledge. Arduino Uno was released in 2010. With Arduino Uno, physical information can be obtained from various sensors, and various experiments can be done with this information. In addition, an output can be obtained from parts such as motor, LED. A basic programming knowledge is sufficient to control such electronic components by connecting them to the Arduino Uno board. The level of electronic and programming knowledge required according to the level of the projects will also increase. Although there are much smaller and much larger models in size, the size of Arduino Uno is the most standard according to projects. Having 14 digital output pins means that 14 different digital sensors and parts can be controlled. This is enough for many projects. 5 of these digital outputs are PWM outputs. Parts that are required to be controlled analogously, such as the speed of the motors and the brightness levels in the LEDs, are controlled by connecting to these PWM pins. 6 analog inputs on Arduino Uno are for sensors that can receive analog input signals. Arduino Uno contains all the necessary components to support a microcontroller. It can be powered by connecting the Arduino Uno to a computer, using an adapter or a battery. The parts of Arduino Uno R3 are shown in Figure 3.1.

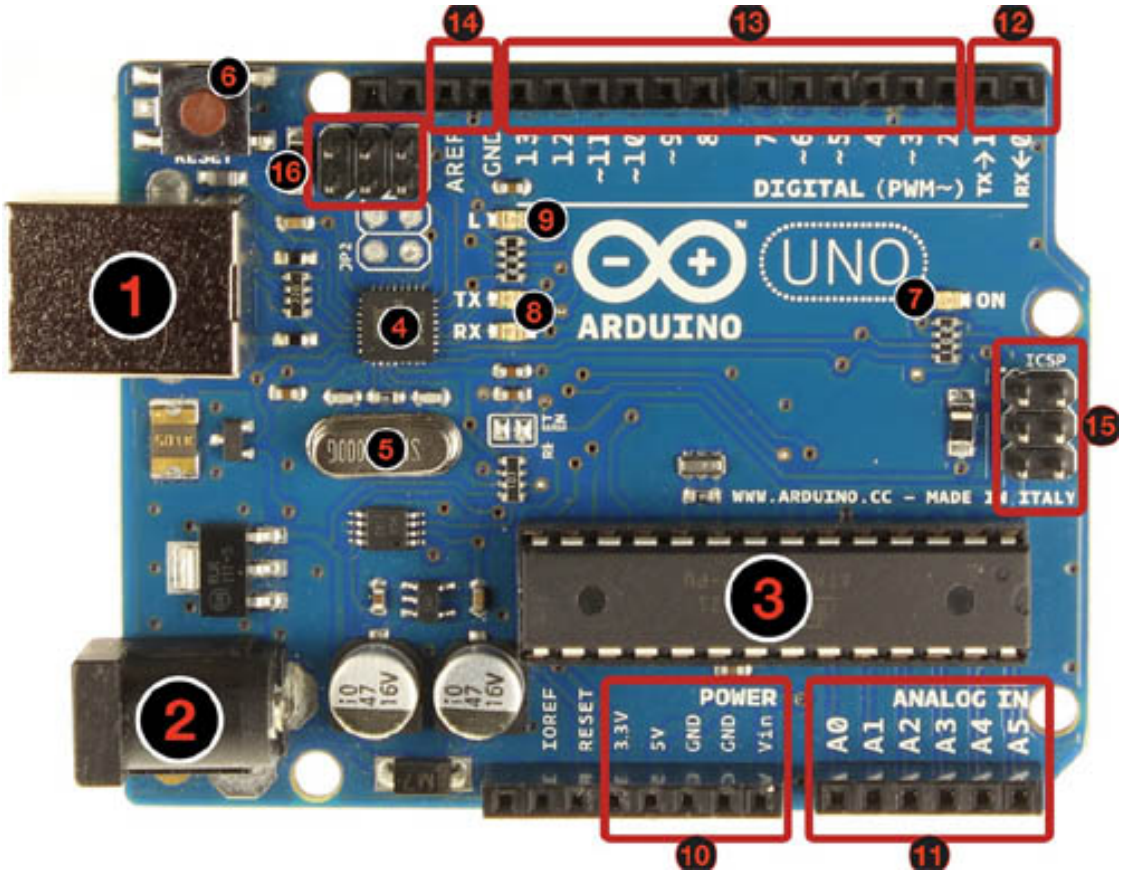


Figure 11 The Part of Arduino Uno

Table 3 List of the Part of Arduino Uno

#	Part Name
1	USB Input
2	Power Input
3	Microcontroller ATmega328
4	Communucation Chip
5	16 MHz Crystal
6	Reset Button
7	Power LED
8	TX / NX LEDs
9	LED
10	Power Pins
11	Analog Inputs
12	TX / RX Pins
13	Digital I/O Pins
14	GRD and AREF Pins
15	ICSP for ATmega328
16	ICSP for USB Interface

2 Arduino Uno were used in the mechanical system. one runs the 3 stepper motors. The other is used to run DC Motor and accelerometer.

2.2.2.2. Stepper Motor

A stepper motor is a brushless, synchronous electric motor that converts digital pulses into mechanical shaft rotation. Its normal shaft motion consists of discrete angular movements of essentially uniform magnitude when driven from sequentially switched DC power supply. The stepper motor is a digital input-output device. It is particularly well suited to the type of application where control signals appear as digital pulses rather than analog voltages. One digital pulse to a stepper motor drive or translator causes the motor to increment one precise angle of motion. As the digital pulses increase in frequency, the step movement changes into continuous rotation. Some industrial and scientific applications of stepper motors include robotics, machine tools, pick and place machines, automated wire cutting and wire bonding machines, and even precise fluid control devices. NEMA17 Stepper Motors are used to

provide movement of the mechanical system. These stepper motors must take 200 steps to make a full turn, so the step angle is 1.8 degrees. Depending on the frequency, changes in speed can be observed.

Figure 12 Nema 17HS3001 Stepper Motor

2.2.2.3. Stepper Motor Driver

Step motor drivers are needed to drive step motors. Basically, the task of the drivers is to keep the motors running. There are many stepper motor drivers, but the A4988 stepper motor driver has been selected. 3 step motors are used for the movement of the mechanical system. In order to operate them independently of each other, 3 step motor drivers are required. In the installation process, the cables of the stepper motors are connected to the driver. Arduino uno connection is established from the driver.

Figure 13 A4988 Stepper Motor Driver

2.2.2.4. Vibration DC Motor

The simplest and most popular type of vibration motor is known as ERM (Eccentric rotary mass vibration motor). It is one of the commonly used DC motors. When the motor runs, an unbalanced centripetal force occurs due to the part at the end while the shaft is rotating. Due to this imbalance, a displacement movement will occur in the motor. Its speed can be adjusted according to its frequency. These displacements are called vibration, and this is the basic logic of vibration DC motors. Vibrating DC motor with model number WRK-450SH was used in the mechanical system.

Figure 14 WRK450SH Vibration DC Motor

2.2.2.5. Accelerometer

The ability of a system to withstand shock and vibration is measured by the system's $'g'$ level. Accelerometers are used to measure this level. Accelerometer is a sensor that measures physical acceleration due to mechanical excitation or inertia force. Acceleration is the change in speed over time. It increases according to how fast the speed changes. A vector with direction and amplitude is a magnitude. Accelerometer, It can be used to measure vibration, rotation, tilt, collision (shock) and gravity. The unit of measurement g is $9.81\ m\ /\ s^2$, which is the acceleration value of gravity. In the project, accelerometer with model number ADXL345 was used. It is very useful and economical.

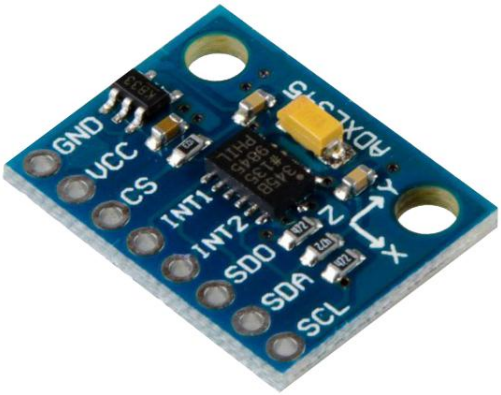


Figure 15 ADXL345 Accelerometer

2.2.2.6. MOSFET

A metal oxide semiconductor field effect transistor (MOSFET, MOS-FET, or MOS FET) is a field effect transistor (FET with insulated gate) in which the voltage determines the conductivity of the device. It is used to change or amplify signals. The ability to vary the conductivity with the amount of applied voltage can be used to amplify or switch electronic signals. A MOSFET is by far the most common transistor in digital circuits because hundreds of thousands or millions of them can be incorporated into a memory chip or microprocessor. The purpose of use of MOSFET in the project is to control voltage and current flow. It is useful for driving DC motor. It helps to use the potentiometer more effectively.

Figure 16 MOSFET

2.2.2.7. Power Supply

Power supply is the part that provides electrical energy to electronic devices that require electrical power. The incoming energy is transferred to the devices in sections. The incoming current comes in the form of 220 V as AC. The power supply converts electrical energy to DC current and distributes the energy required for the mechanical system. Power supply also prevents power surges. Protects the electronic assembly against voltage fluctuations. It differs according to the electrical energy supplied. Since there is 1 DC motor and 3 Stepper motors in this project, a power supply of 12 V, 1.6 A and 220 W has been preferred. Without power supply, it is not possible to keep the mechanical system moving because Arduino can only supply 5 V voltage to the electronics.

Figure 17 12 V, 6 A, 220 W Power Supply

2.3. Assembly and Wiring

The parts drawn in this section will be assembled. During the design of the mechanical system, 3 stepper motors and 1 vibration DC motor were used. Therefore, there should be 3 carrier plates. Stepper motors can be fixed on this plate. Assembly phase consists of 2 parts. First, carrier plate and M12 square nut were mated and the degree of freedom to move back and forth on the surface was given. This freedom is given to allow the spring to be stretched during the operation of the stepper motors. Top units have been added to the bottom of the carrier plate. This part is not included in the assembly because it was attached to the carrier plate during drawing. It is shown in Figure 2.11. Installation of the part where stepper motors will be fixed has been completed. Next is the assembly of all the parts.

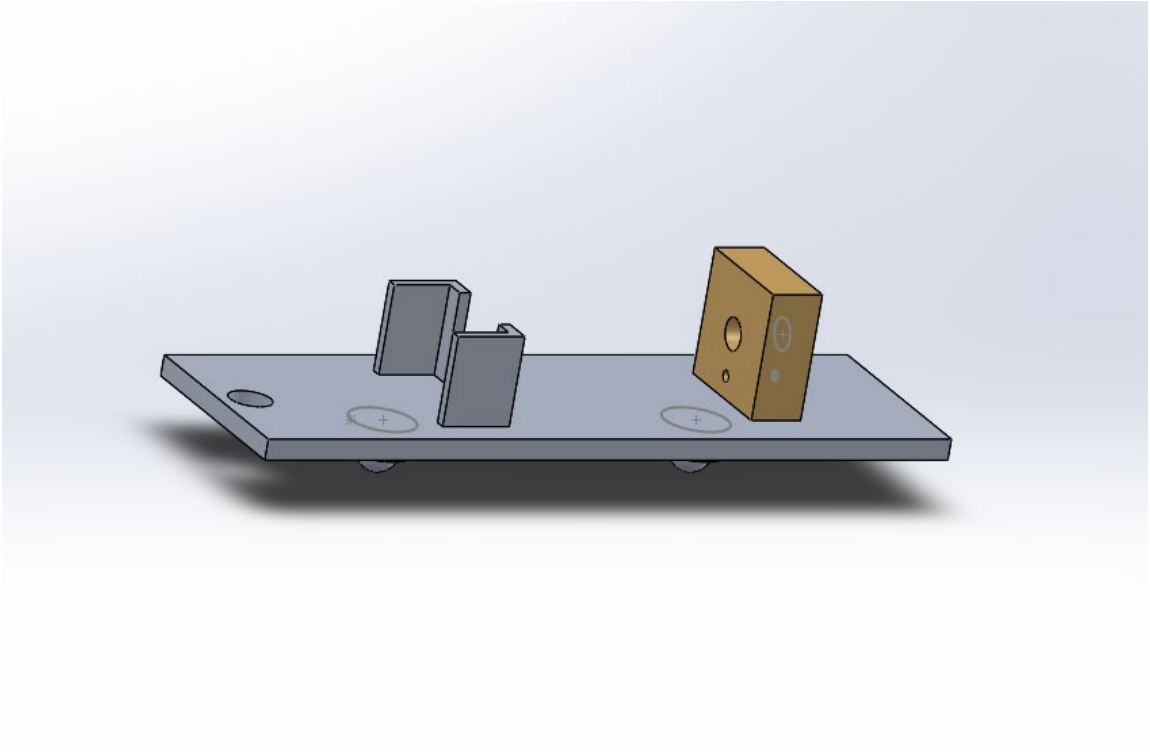


Figure 18 Assembly for Carrier Plate, M12 Square Nut and Ball Unit

In the mechanical system, freedom is given movement of the carrier plate. Under normal conditions, the main plate and the carrier plate are fixed with M14 pivots. But during the assembly, 2 parts were assembled by mate them. Then the vibration plate and the main plate should be mounted. Thrust bearing was placed between the main plate and the vibration plate. This part will facilitate the movement of the vibration plate on the main plate. Besides, possible abrasions and unnecessary energy loss are prevented. It is shown in Figure 2.12. Thus, the assembly of the mechanical part is completed. Now, the part related to the introduction and properties of electronic parts will be passed.

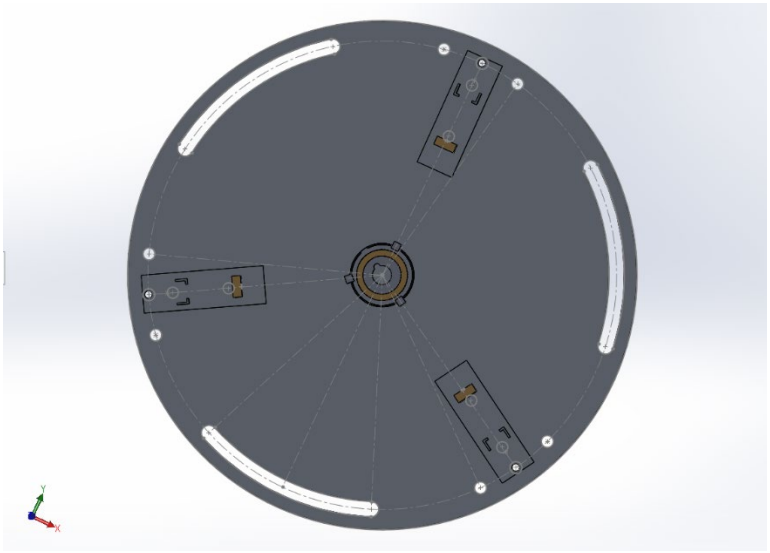


Figure 19 Assembly for Main Plate, Vibration Plate, Carrier Plates, Thrust Bearing

How the electronic parts of the mechanical system are connected will be explained. While the circuit was established, many electronic parts were used, and a lot of cables were used for their connection. The coding part will be explained after the installation. Basically, the system will consist of 2 electronic circuits. The reason for the establishment of 2 separate circuits is to understand the accelerometer data more accurately. In addition, it is aimed to reduce the heating problems of stepper motors. The first circuit is the circuit in which elements such as Vibration DC motor, Potentiometer are used as shown in Figure 3.12. The second circuit is the circuit with elements such as stepper motors, LCD screen, and Joystick module, as shown in Figure 3.13. Breadboard was preferred because there is too much wiring. It is shown in Figure 3.11.

Figure 20 Breadboard

A program called Fritzing was used to create the circuit diagrams. Fritzing is a very useful electronic program for creating circuit diagrams. Since it is an open-source program, it is very easy to access. It is a program where everyone can model the circuit as they wish. Which pins are used in wiring is one of the most important issues. Because it is very important to define the correct pins while defining the pins in the coding part, otherwise the code will fail. The software part of the work is also very important.

Table 4 List of Setup with Arduino for Vibration DC Motor

#	Item
1	Arduino Uno
2	Power Input
3	Power Swich
4	Vibration DC Motor
5	MOSFET
6	Accelerometer
7	Potantiometer

Table 5 List of Setup with Arduino for Stepper Motor

#	Item
1	Arduino Uno
2	Stepper Motors
3	Stepper Motor Drivers
4	Power Swich
5	Power Input

6	I2C 16x2 LCD Screen
7	Joystick Module

2.4. Coding

We will use stepper motors to activate our actuators. We will command our stepper motors using the Arduino IDE. Arduino is an open-source electronic platform based on easy-to-use hardware and software. Arduino, which has started to become popular especially among those dealing with technology and electronics, is preferred by many people as a hobby.

This flexibility, combined with the fact that the Arduino software is free, hardware boards are inexpensive, and both software and hardware are easy to learn has led to a large community of users contributing codes and posting instructions for a wide variety of software.

Arduino editor page consists of 2 parts. These are the setup and loop parts. This is the part where we enter the information about our coding in the setup section. The loop part is the part where we create the code algorithm and shape it as we want.

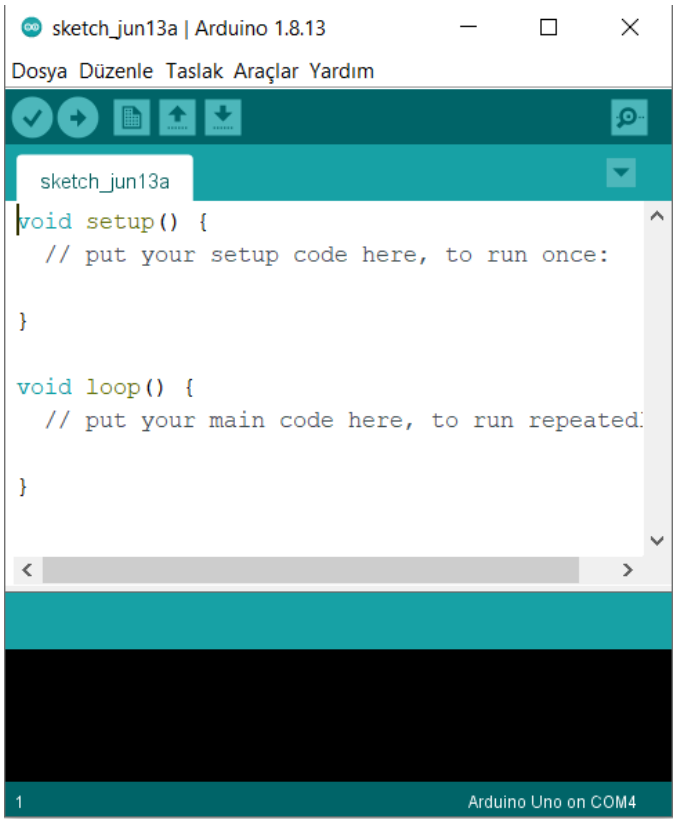


Figure 21 Arduino Interfac

Before we write code in Arduino, we must define which library we choose. we use stepper motor and dc motor libraries in our codes. Although we do not use a library, the commands do not work and give an error.

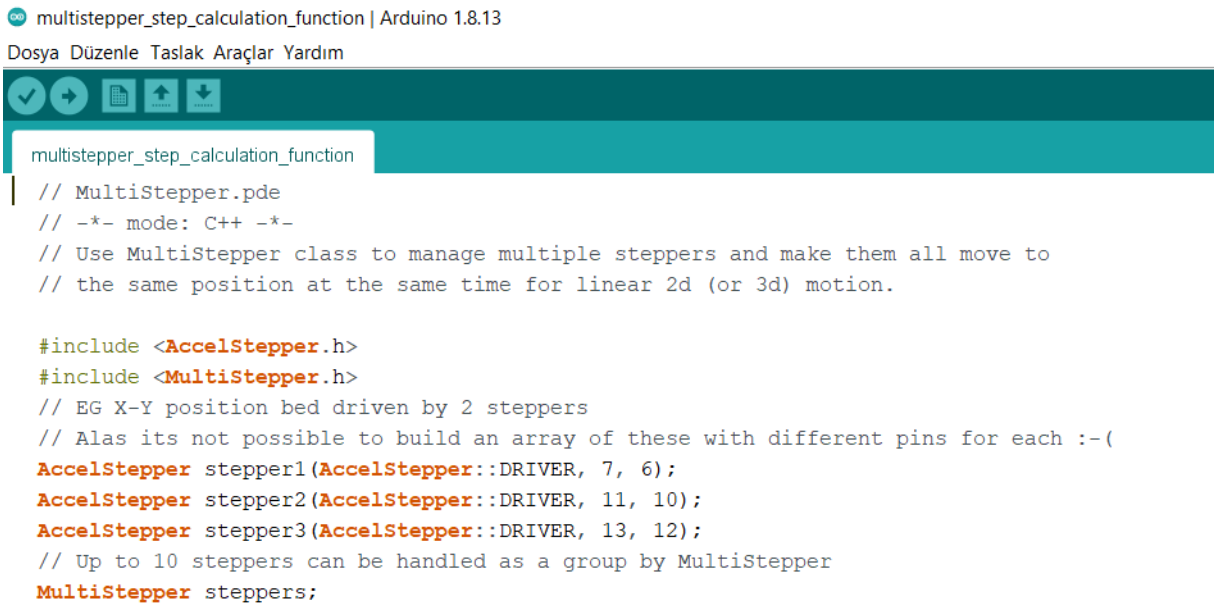


Figure 22 Arduino Definition of Library

We should explain the basic logic of our coding. Our aim is to control the stiffness at a point we want at the intervals we have determined. For this, we need to run our stepper motors at different values. our code ensures that the stepper motors start and stop at the same time. Since we enter different values, it is not possible for us to run at the same speed, because if they run at the same speeds, they cannot be stopped at the same time. As a solution to this, we determined the maximum speed. The code adjusts the speed of the stepper motors according to the value entered and allows them to work simultaneously. Enter the commands as shown below.

```
//Configure each stepper

stepper1.setMaxSpeed(200);                // The max speed for Stepper Motor 1
stepper2.setMaxSpeed(200);                // The max speed for Stepper Motor 2
stepper3.setMaxSpeed(200);                // The max speed for Stepper Motor 3

//Then give them to MultiStepper to manage

steppers.addStepper(stepper1);            // The command which start and finish at the same time for Stepper Motor 1
steppers.addStepper(stepper2);            // The command which start and finish at the same time for Stepper Motor 2
steppers.addStepper(stepper3);            // The command which start and finish at the same time for Stepper Motor 3
```

Figure 23 Arduino Definition of Maximum Speed

A large part of our coding consists of calculations. Position and stiffness checks should be made according to the results. To make it simpler, we created 2 functions different from the setup. It is shown in more detail in the appendix.

3. RESULT & DISCUSSION

After we make some tests, we obtained that we can really manipulate the stiffness of the end-effector at E point. We take high frame videos to observe that difference. As we rotate the DC motor, the vibration plate starts to oscillate and has an amplitude of displacement.

Before we talk about tests, we want to talk about calculations that is required for the tests. 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. We have,

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

If we derivate the equation above it yields,

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

In our experiments we have used spring sets that has maximum k =539 N/m.

We calculated k's from Newton's Law.

$$F = ma$$

We hang a 5 kg (kilogram) weight on the spring and measured the displacement it made from its original unstretched length. Original length of the spring is 162 mm. After the weight, length difference of the spring was 91 mm. Since the gravitational acceleration on earth $g = 9.81\ m/s2$,

$$49.05\ N = k\ (0.091\ m)$$

$$\boldsymbol{k = 539\ N/m}$$

After these calculations we are ready to start tests.

In first test, we selected our fi as 1000 and we moved the end effector to the point (1,1). After we moved the end effector to the point (1,1) with fi = 1000. We run our dc motor which is on the end effector with 50 rev/min. From the accelerometer we got the acceleration data for x, y and z direction of end effector. From the accelerometer we got 200 data per second. We collected data for 10 seconds long.

We loaded these data to the MATLAB program to plot the accelerations. Before we plot these acceleration data we filtered the data, because data can be much higher or much lower than usual for some points. With this way we hoped to get more stabilize graphs.

After the filtering we plotted Ax and Ay by time. The results are as follows.

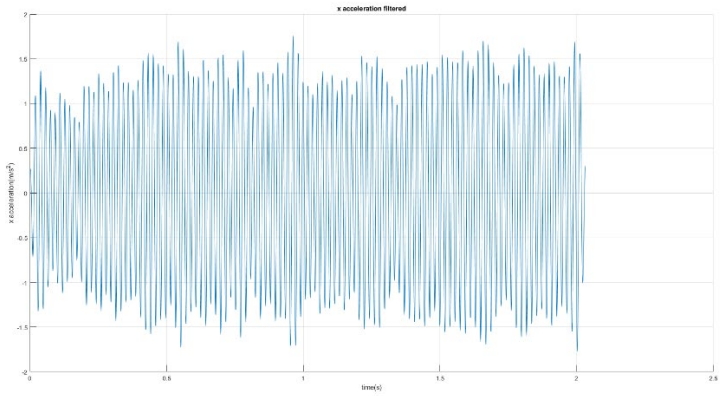


Figure 24 Ax vs Time Fi=2000

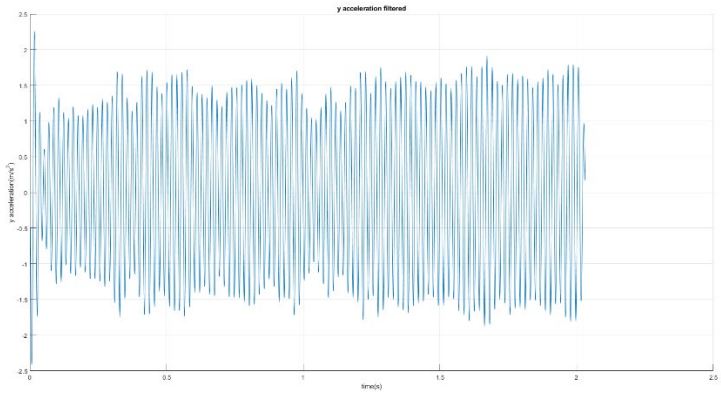


Figure 25 Ay vs Time Fi=2000

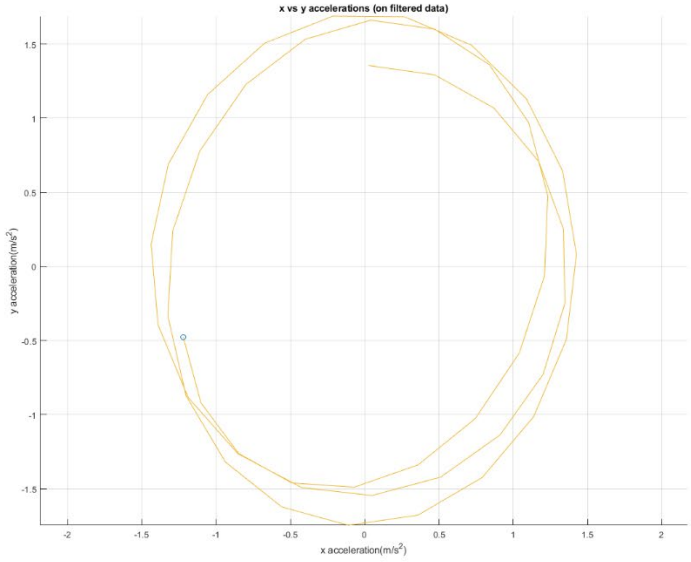


Figure 26 Ax vs Ay Fi=2000

As you can see from Figure 24 and Figure 25, accelerations of the end effector on x and y directions are not smooth. They are increasing by time. This may be caused by the accelerometer we have used for this test. Or this may be caused by the DC Motor we used. DC Motor could not be running constant.

From the error above we got Ax vs Ay graph like Figure 26. This graph draws ellipses which are going bigger by time. We could not prevent the errors that we got from Figure 24 and Figure 25. We decided to use the smallest ellipse that we can observe from Figure 26 to compare the tests.

The graph that we used to compare with others shown down below.

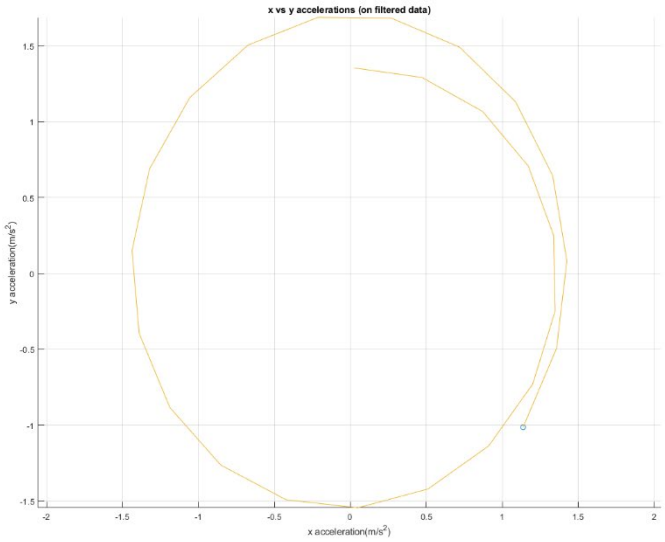


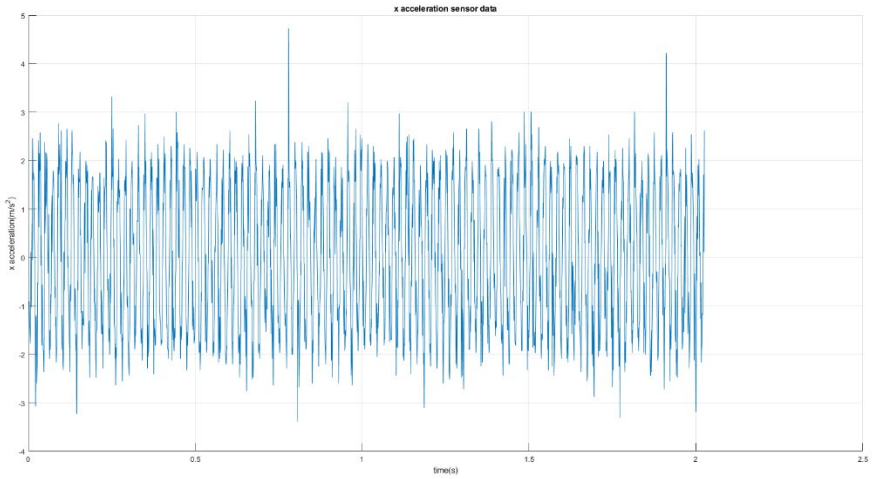
Figure 27 Ax vs Ay Simplified Fi=2000

The ellipse we see on Figure 27 represents as Stiffness Ellipse. Stiffness ellipse shows us the rigidness of the end effector on x and y directions for specific circumstances that we used for this test.

From this graph radius of this ellipse on x direction is approximately 1.49 m/s^2. And the radius on y direction is approximately 1.75 m/s^2.

These radiuses show us the end effector is stiffer on x direction.

In the second test, we keep the position as (1,1) but we increased fi value to 3000. After using the same rules, the results we get are as follows.



+

Figure 28 Ax vs Time Fi=3000

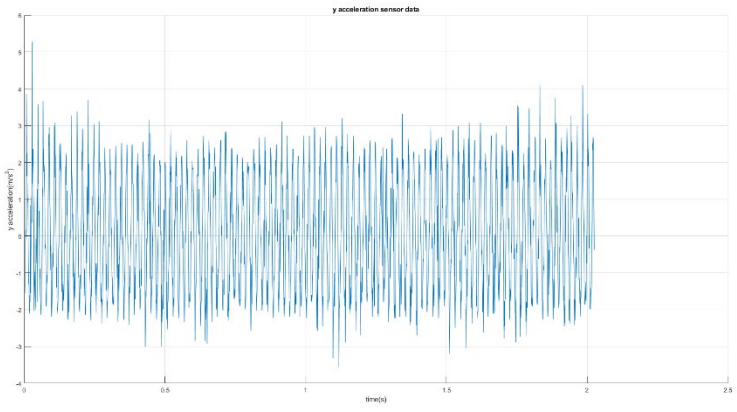


Figure 29 Ay vs Time Fi=3000

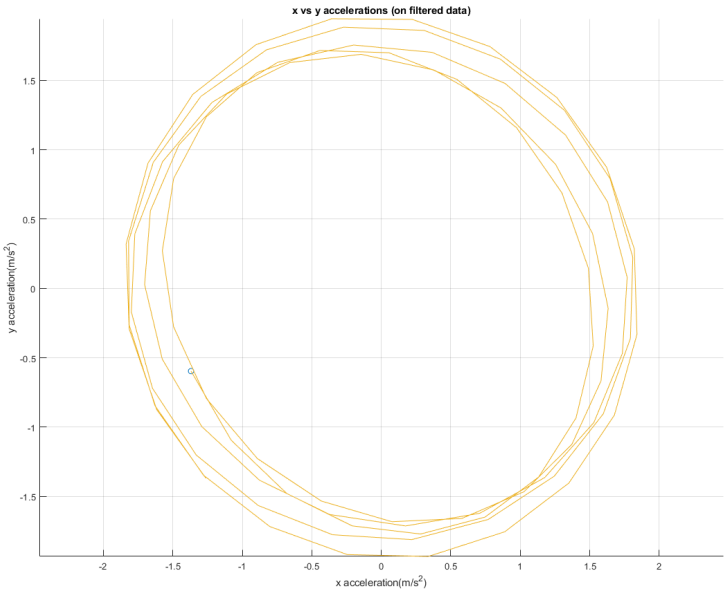


Figure 30 Ax vs Ay Fi=3000

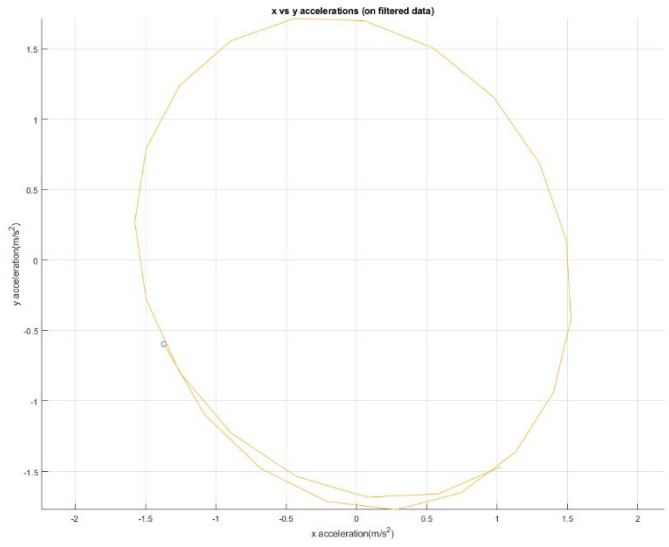


Figure 31 Ax vs Ay Fi=3000

If we look at the Figure 31, we can see that our ellipse rotated. This may be caused by the rotation of the end effector due to movement.

The ellipse has radius on x direction as approximately 1.47 m/s^2 and the radius on y direction is approximately 1.72 m/s^2.

From these values if we want to compare two tests, we can say that the stiffness ellipse on second test is smaller that the first one. This means we have managed to alter the stiffness of the end effector.

Values are very small. Because of that the change in stiffness ellipses are very small. We can not observe it by eye. This may be caused by the unbalanced load provided by our dc motor. The unbalanced load we applied to end effector is small. Because of that we can not observe the result by eye.

4. CONCLUSION

In this project, we have learned how to manage an engineering project process. Also get a lot acknowledge about manufacturing process, Arduino coding and electrical components. We had a lot of problems during project about the mechanical and electronic design. We have tried to solve these problems. These problems have costed our time a lot. But at the end, it was so helpful to us for learning.

REFERENCES

[1] Inoue,H., Tsusaka,Y., Fukuizumi,T., "Parallel manipulator", Proc. 3rd International Symposium on Robotics Research, MIT Press Series in Artificial Intelligence, Ch7, pp.321-327, **1986**

[2] Wang,Y.X., Wang,Y.M., "Inverse kinematics of variable geometry parallel manipulator", Mechanism and Machine Theory, v.40, p.141-155, **2005**

[3] Naccarato,F., "Kinematics of Variable Geometry Trusses", Ph.D. Dissertation, Institute for Aerospace Studies, Univ. of Toronto, **1995**

[4] Dong,W., Du,Z., Sun,L., "A large workspace macro-micro dual parallel mechanism with wide-range flexure hinges", IEEE International Conference on Mechatronics & Automation, pp.1592-1597, July,**2005**

[5] Cho,Tesar,Freeman, "The dynamic and stiffness modeling of general robotic manipulator systems with antagonistic actuation", IEEE Proc. Intl Conf on Robotics and Automation, v.3, pp.1380-1387, **1989**

[6] Rong,L., Guanghua,Z., "Dynamics of Parallel Mechanism with Direct Compliance Control", IEEE Systems, Man, and Cybernetics, 'Computational Cybernetics and Simulation', v.2, pp.1753-1758, **1997**

[7] Alici,G., Shirinzadeh,B., "Enhanced Stiffness Modeling, Identification and Characterization for Robot Manipulators", IEEE Transactions On Robotics, v.21, n.4, Aug,**2005**

[8] Robert Stawell Ball, "A Treatise on the Theory of Screws", Cambridge Mathematical Library, University of Cambridge, 1900, "http://cdl.library.cornell.edu/cgibin/ cul.math/docviewer?did=03000001&seq=9"

[9] Dimentberg,F.M., "The Screw Calculus and its Applications in Mechanics", Foreign Technology Div Wright-Patterson, Air Force Base, Ohio, Doc.No.:FTD-HT-23-1632-67, **1965**

[10] Loncaric,J., "Geometrical analysis of compliant mechanisms in Robotics," Ph.D. Dissertation, Harvard Univ., Cambridge, MA, **1985**

[11] Roberts,R.G., "A Note on the Normal Form of a Spatial Stiffness Matrix", IEEE Transactions On Robotics And Automation, v.17,n.6, pp.968-972, Dec,**2001**

[12] Bayraktar,H., "Stiffness Control of Redundantly Actuated Variable Geometry Truss Manipulators", Thesis of MSc., Univ.of Marmara, **2002**

[13] Bayraktar,H., "Analtical Deriviation of Stiffness in Redundantly Actuated Planar Parallel Mechanisms", Thesis of Ph.D., Univ.of Marmara, **2016**

APPENDIXES

a) Appendix-I Technical Drawing of System

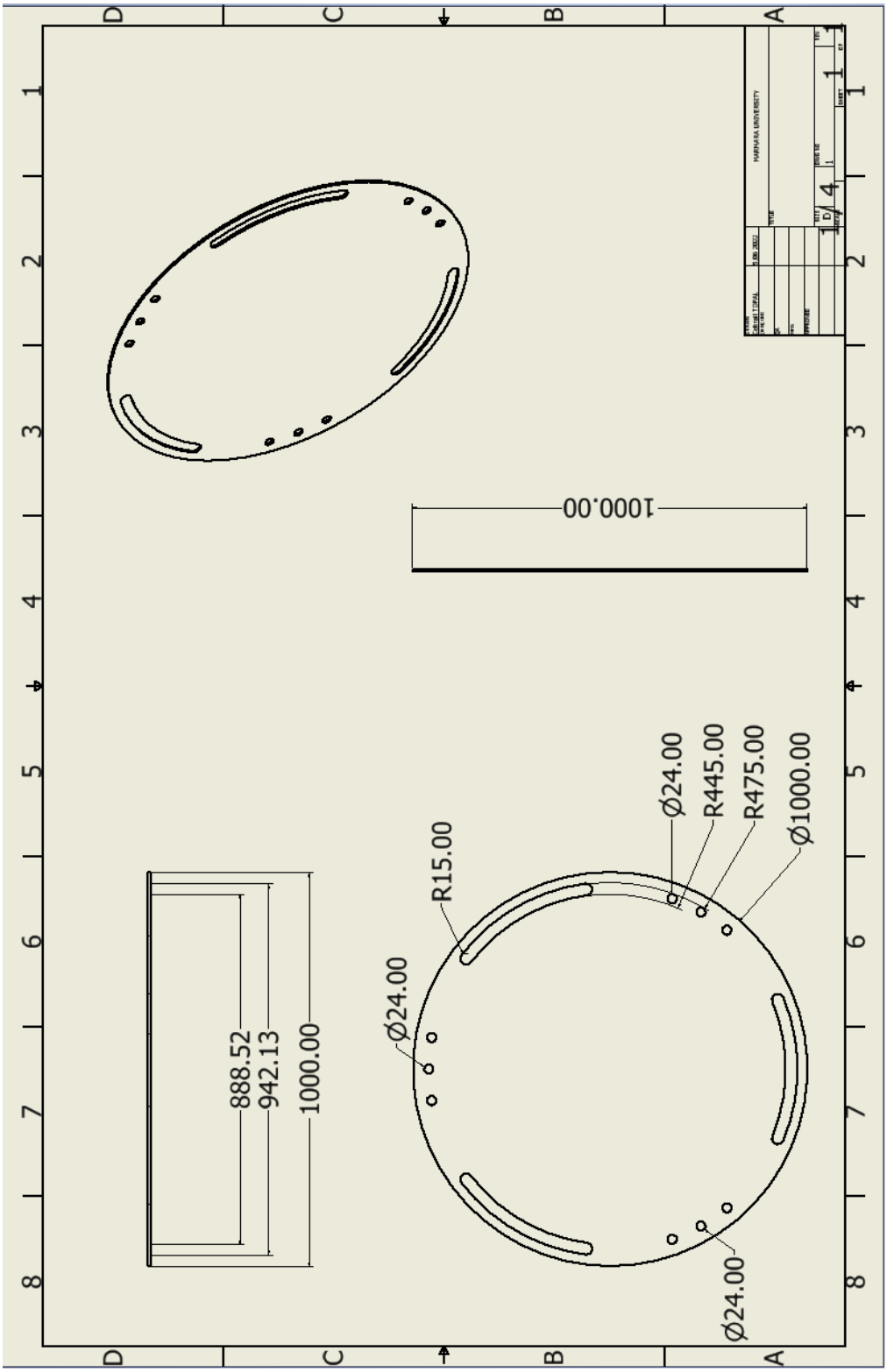


Figure 32 Technical Drawing of Main Plate

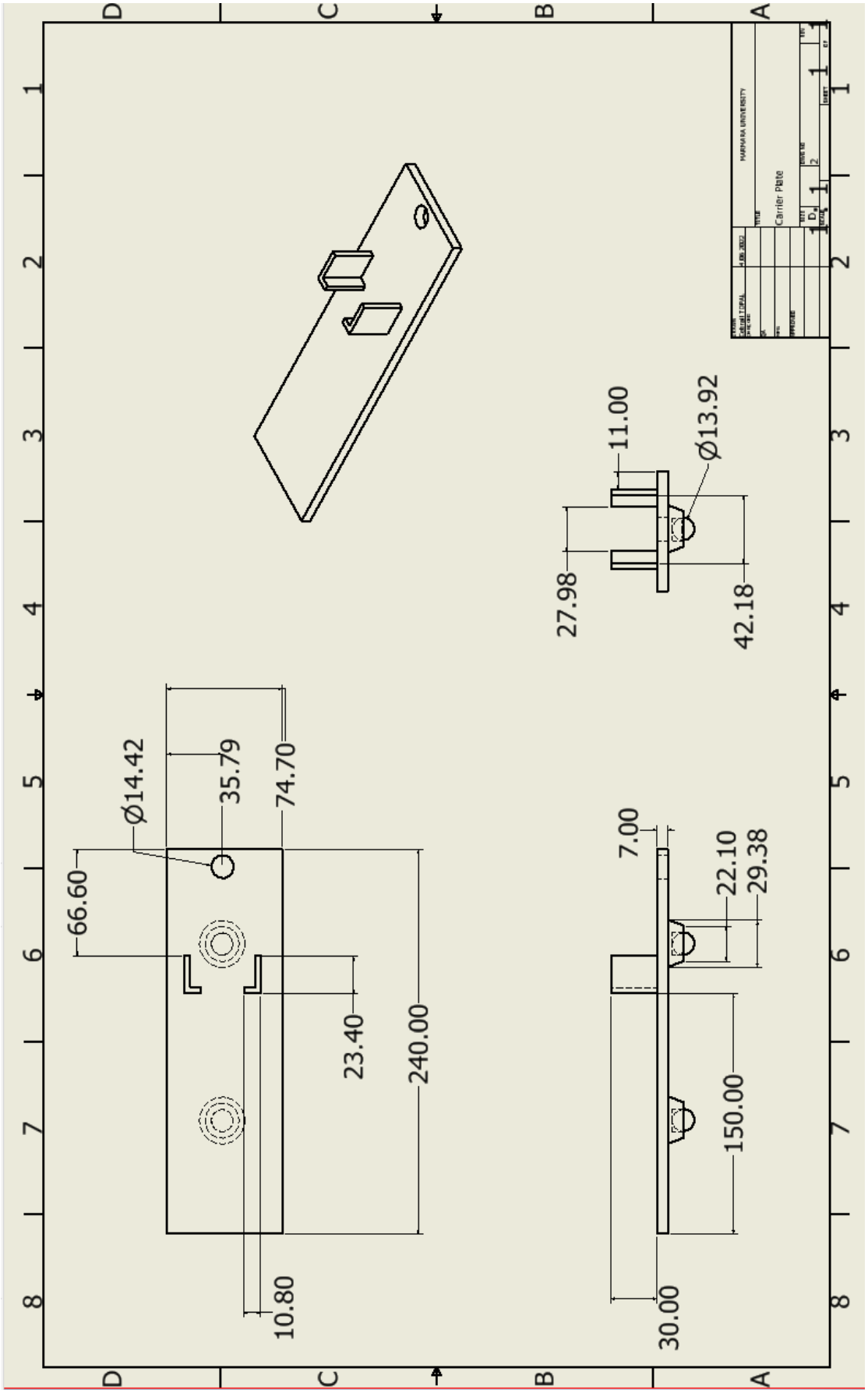


Figure 33 Technical Drawing of Carrier Plate

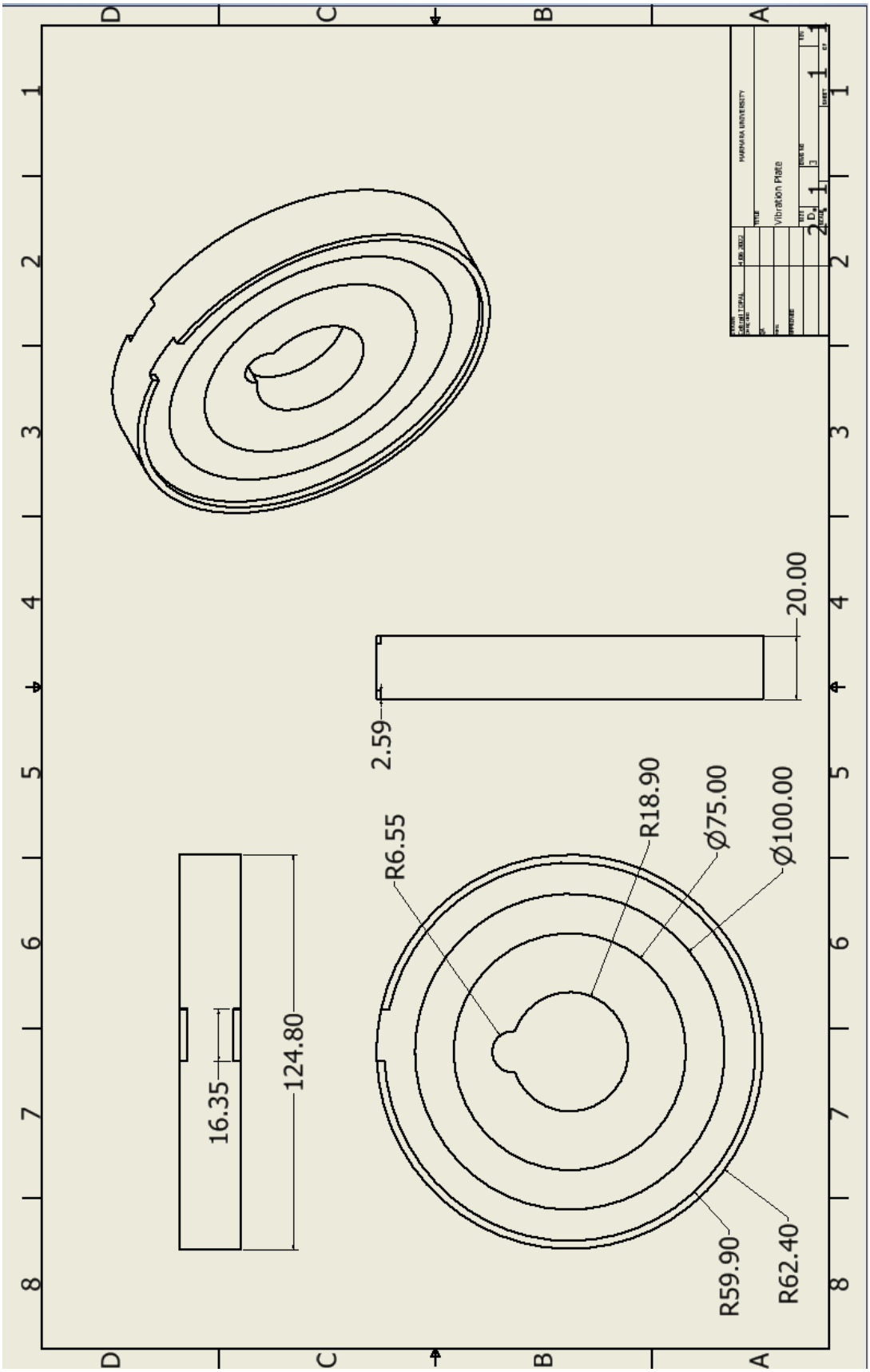


Figure 34 Technical Drawing of Vibration Plate

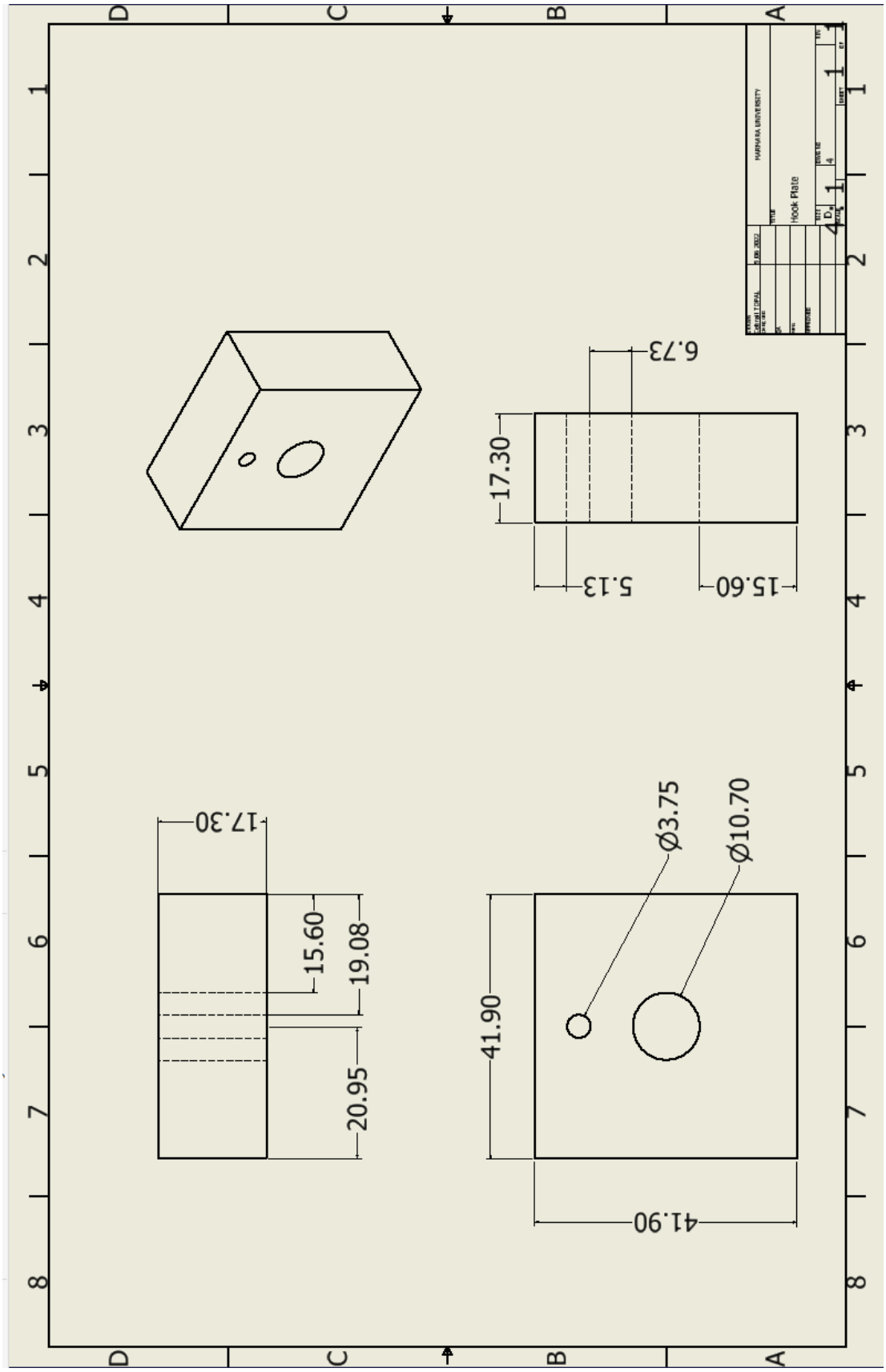


Figure 35 Technical Drawing of Hook Plate

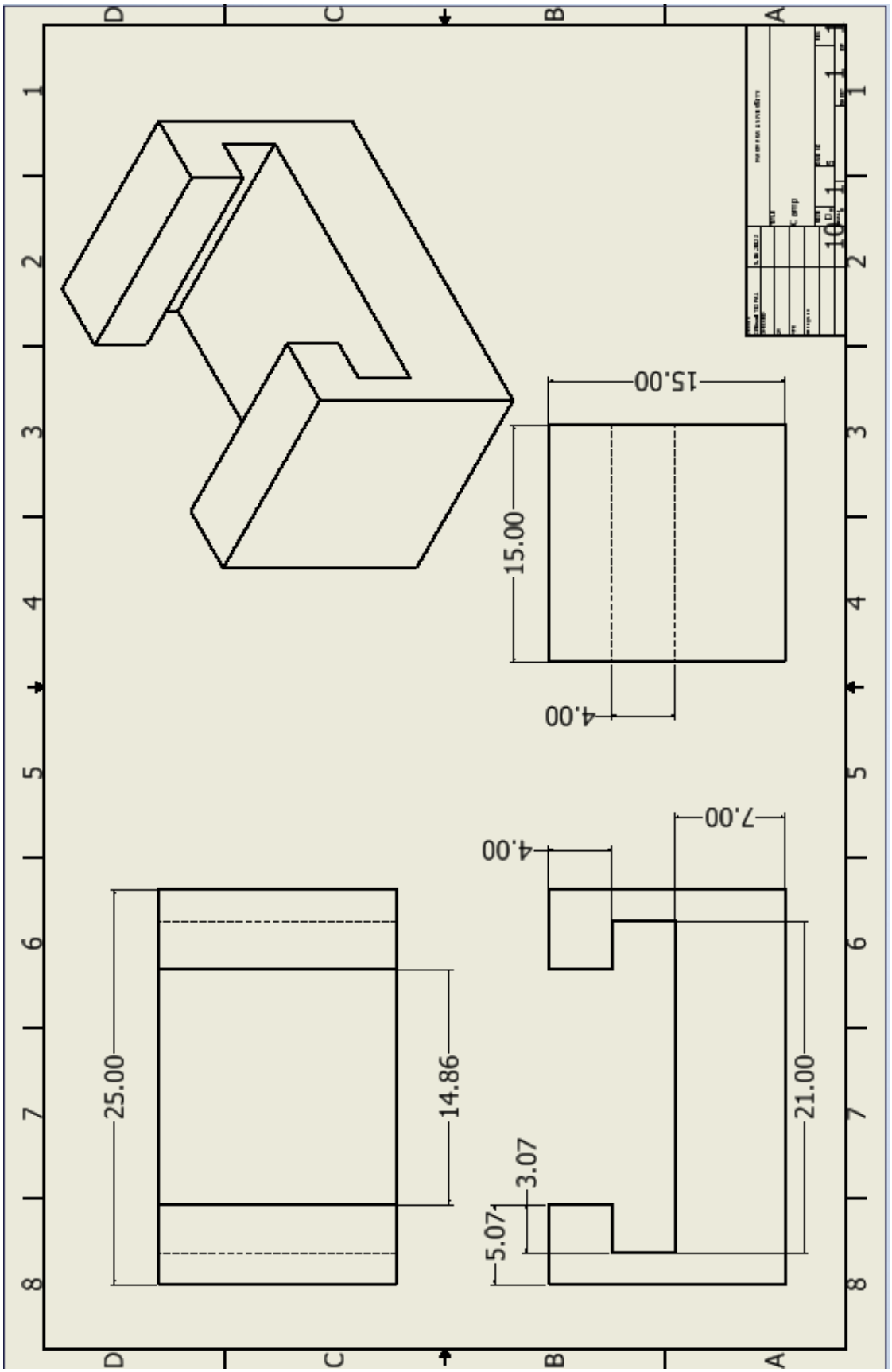


Figure 36 Technical Drawing of Clamp

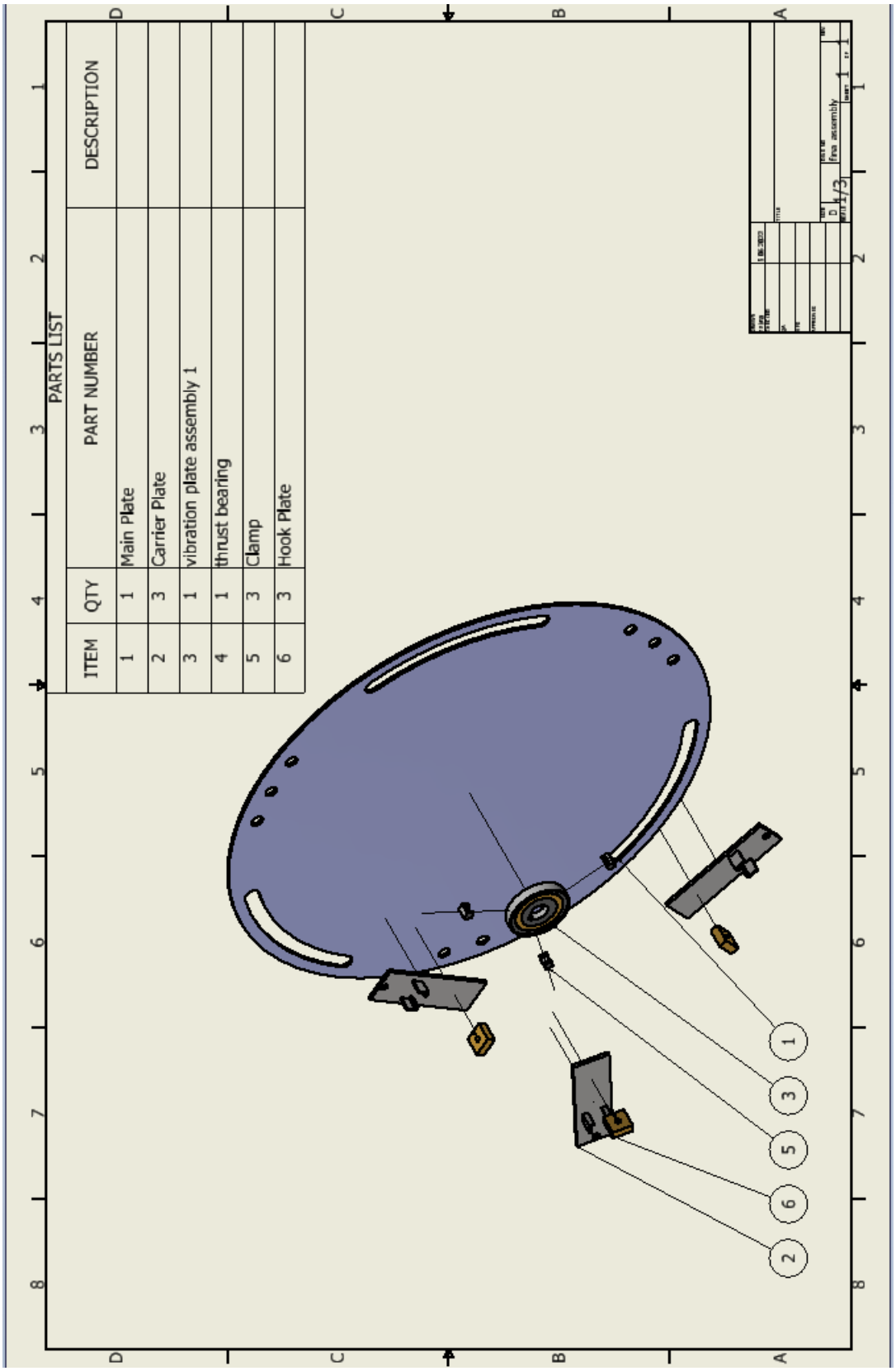


Figure 37 Technical Drawing of System Exploded View

b) Appendix-II Electronic Datasheet of The System

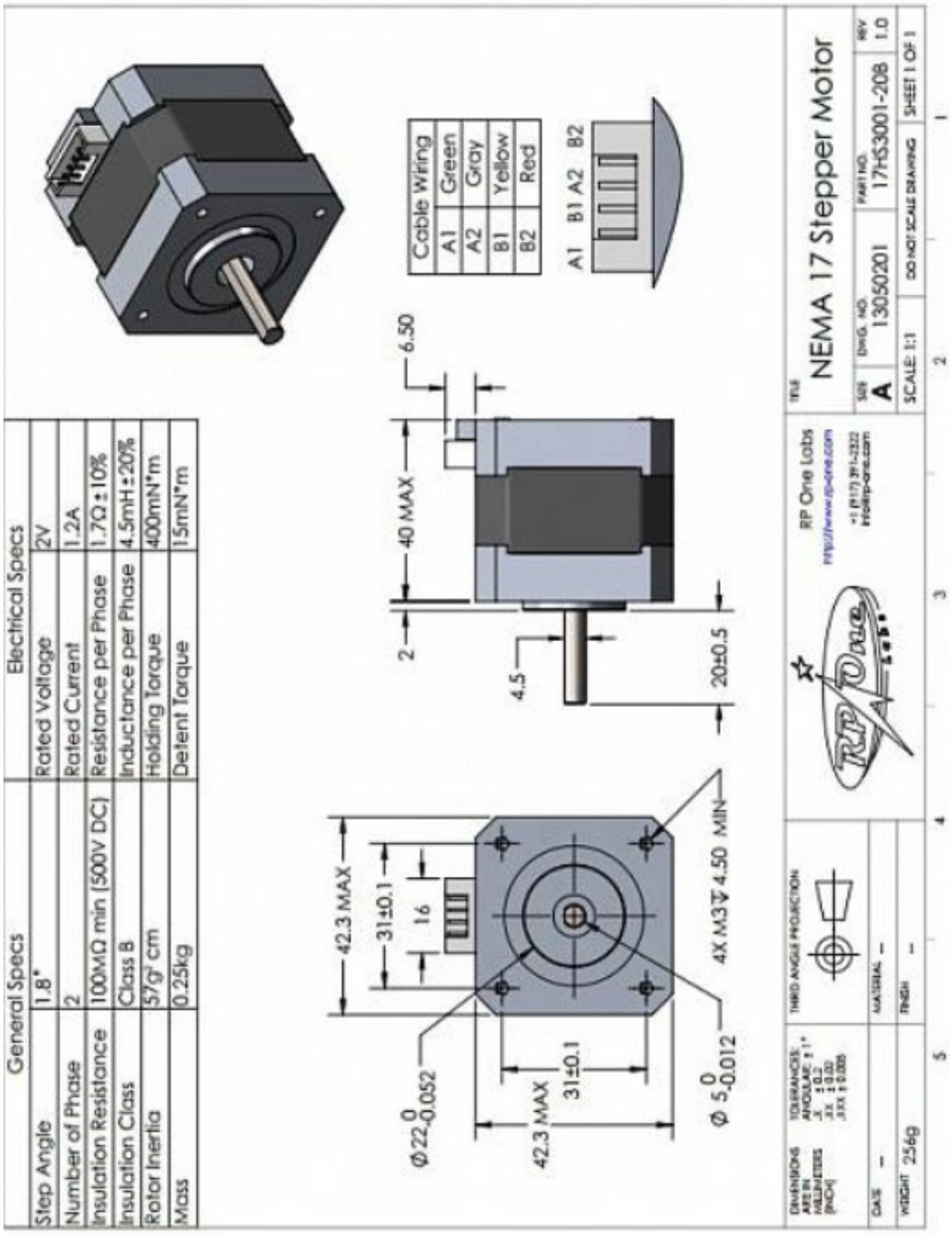


Figure 38 Datasheet of The Nema 17 Stepper Motor

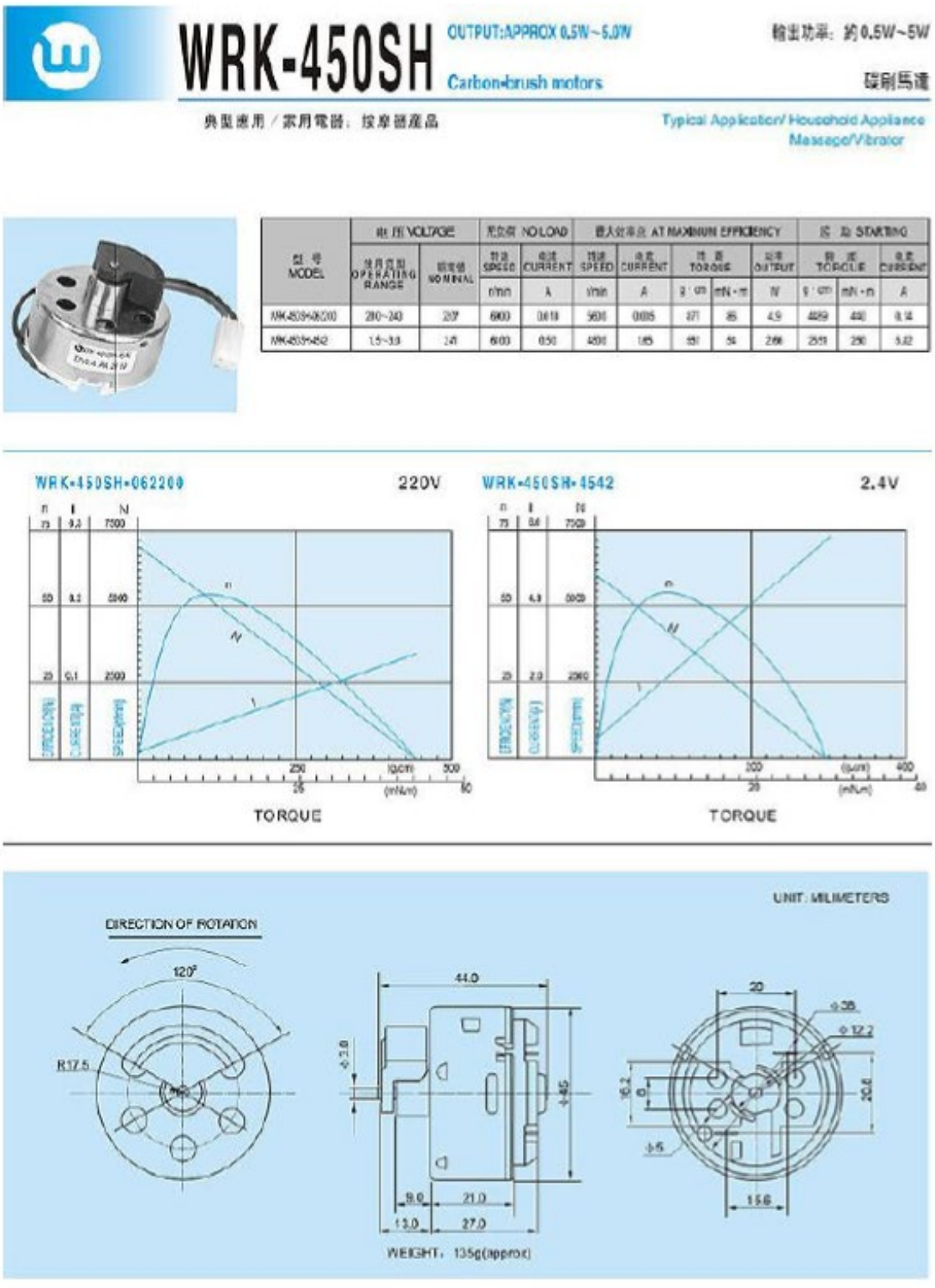


Figure 39 Datasheet of The Vibration DC Motor

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

Figure 40 Technical Specification of The Arduino Uno

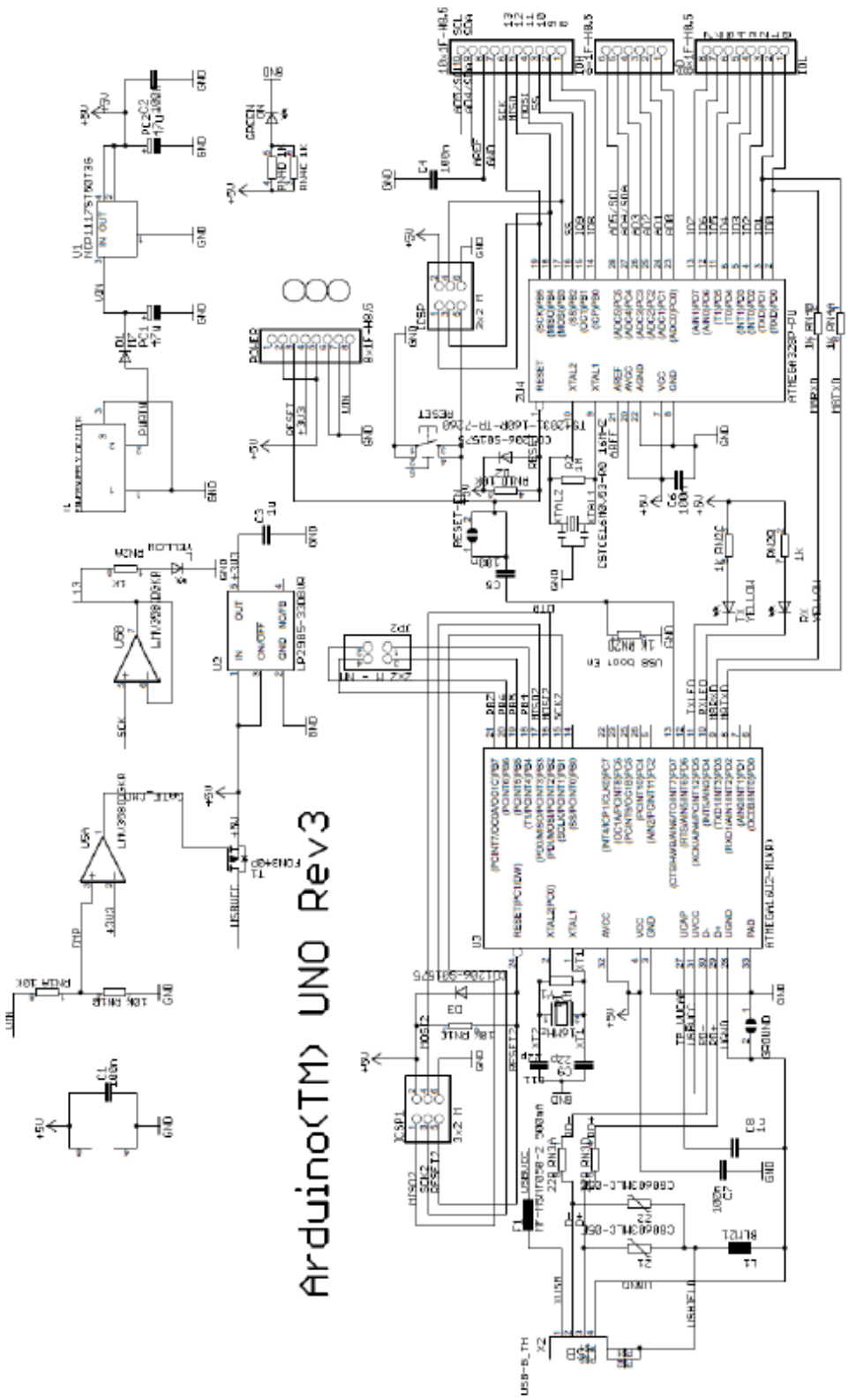


Figure 41 Schematic of The Arduino Uno

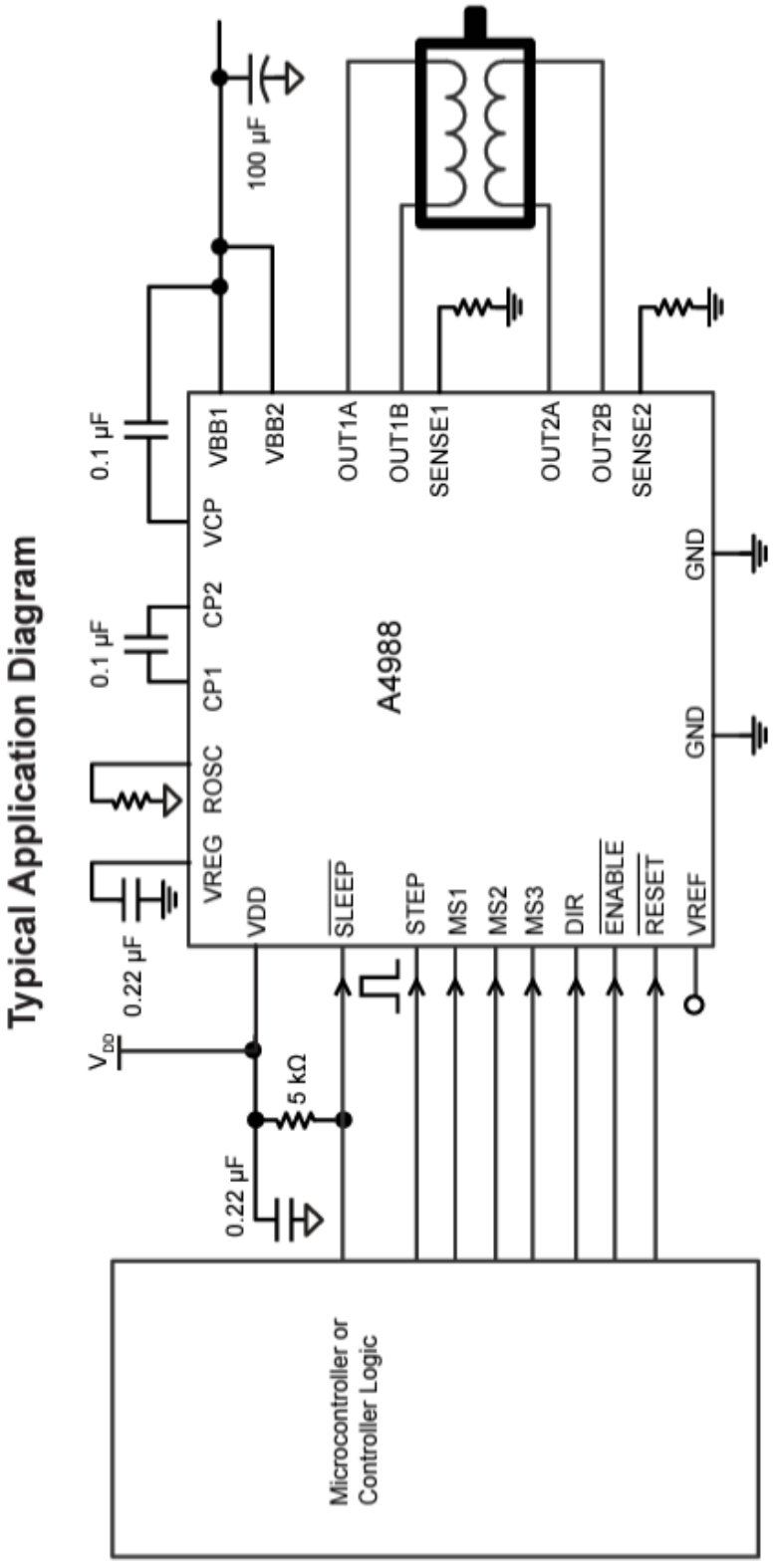


Figure 42 Schematic of The A4988 Driver

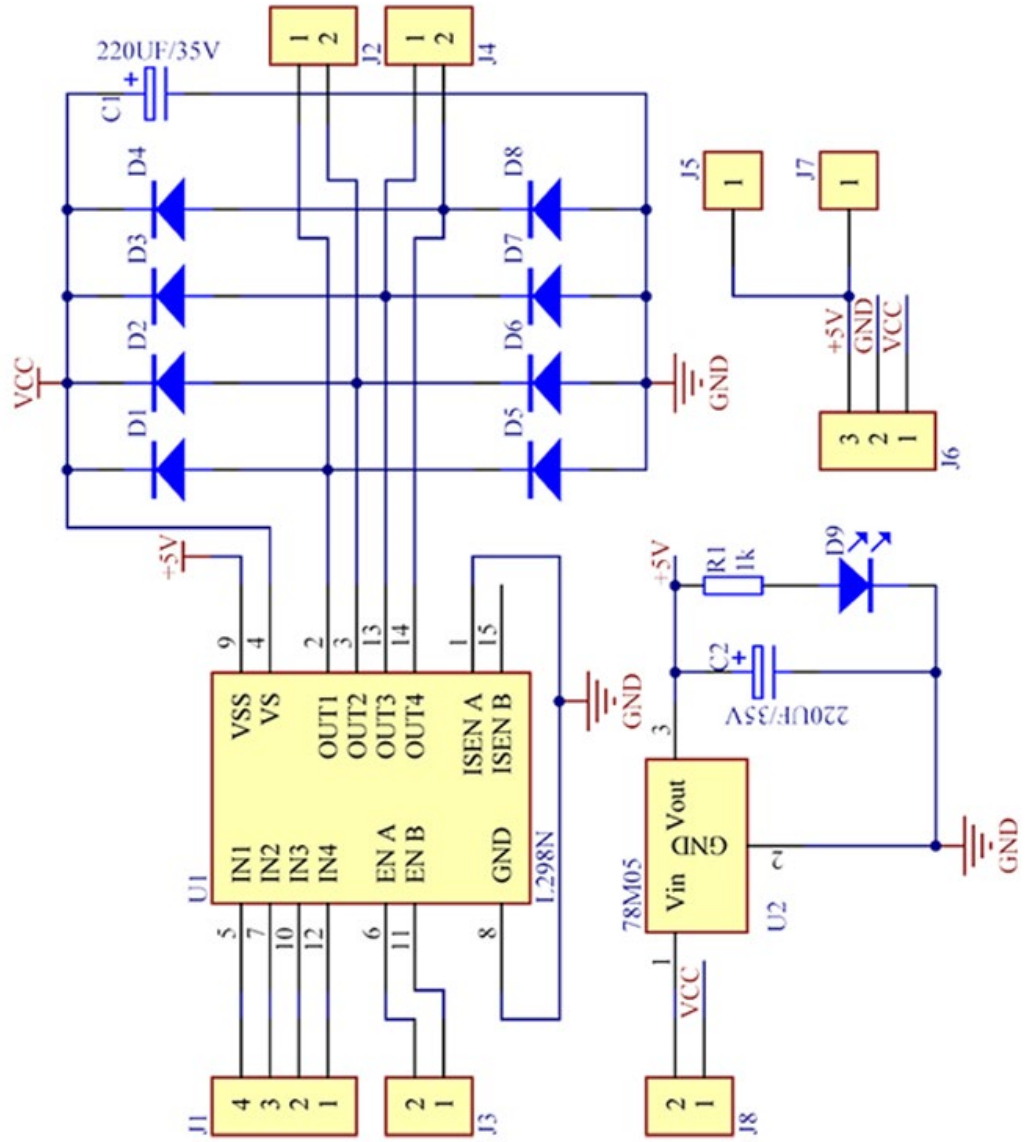


Figure 43 Schematic of The L298N

c) Appendix-III Codding of Stepper Motor

```
// MultiStepper.pde
// -*- mode: C++ -*-
// Use MultiStepper class to manage multiple steppers and make them all move to
// the same position at the same time for linear 2d (or 3d) motion.

#include <AccelStepper.h>
#include <MultiStepper.h>

// EG X-Y position bed driven by 2 steppers
// Alas its not possible to build an array of these with different pins for each :-(

AccelStepper stepper1(AccelStepper::DRIVER, 7, 6);
AccelStepper stepper2(AccelStepper::DRIVER, 11, 10);
AccelStepper stepper3(AccelStepper::DRIVER, 13, 12);

// Up to 10 steppers can be handled as a group by MultiStepper
MultiStepper steppers;

void setup() {

}


 void calculation_1 (float xE, float yE){

  float L0 = 460;
  float x1 = L0;
  float x2 = L0 * cos(120*DEG_TO_RAD);
  float x3 = L0 * cos(240*DEG_TO_RAD);
  float y1 = L0 * sin(0*DEG_TO_RAD);
  float y2 = L0 * sin(120*DEG_TO_RAD);
  float y3 = L0 * sin(240*DEG_TO_RAD);


  float a1 = pow((x1-xE),2);
  float a2 = pow((x2-xE),2);
  float a3 = pow((x3-xE),2);

  float b1 = pow((y1-yE),2);
  float b2 = pow((y2-yE),2);
  float b3 = pow((y3-yE),2);

  float L1 = sqrt(a1 + b1);
  float L2 = sqrt(a2 + b2);
  float L3 = sqrt(a3 + b3);

  float delta_L1 = L1 - L0;
  float delta_L2 = L2 - L0;
  float delta_L3 = L3 - L0;

  float dS1 = (delta_L1 * 200);
  float dS2 = (delta_L2 * 200);
```

```
 float dS3 = (delta_L3 * 200);

 Serial.begin(38400);
 Serial.println(dS1, DEC);
 Serial.println(dS2, DEC);
 Serial.println(dS3, DEC);


 // Configure each stepper
 stepper1.setMaxSpeed(200);   // step/second
 stepper2.setMaxSpeed(200);
 stepper3.setMaxSpeed(200);

 // Then give them to MultiStepper to manage
 steppers.addStepper(stepper1);
 steppers.addStepper(stepper2);
 steppers.addStepper(stepper3);
long positions[3]; // Array of desired stepper positions

 positions[0] = dS1;
 positions[1] = dS2;
 positions[2] = dS3;
 steppers.moveTo(positions);
 steppers.runSpeedToPosition(); // Blocks until all are in position
 delay(2000);

}



 void calculation_2 (float xE, float yE){


 float Kl = 588;
 float c1 = (2*xE + 46);
 float c2 = (3*sq(xE) - 276*xE + 3*sq(yE) + 6348);
 float nom1 = -c1 * sqrt(c2);

 float c3 = (3*yE + sqrt(3)*xE - 46*sqrt(3));
 float c4 = (sq(xE) + 46*xE + sq(yE) + 46*sqrt(3)*yE + 2116);

 float denom1 = c3 * sqrt(c4);

 float No1 = nom1/denom1;

 float c5 = 3*yE - sqrt(3)*xE + 46*sqrt(3);
 float c6 = sq(xE) + 46*xE + sq(yE) - 46*sqrt(3)*yE + 2116;

 float nom2 = -c5 * sqrt(c6);

 float c7 = 3*yE + sqrt(3)*xE - 46*sqrt(3);
 float c8 = sq(xE) + 46*xE + sq(yE) + 46*sqrt(3)*yE +2116;

 float denom2 = c7*sqrt(c8);
```

```
float No2 = nom2/denom2;

float No3 = 1;

float u1 = (xE - 46)/sqrt((sq(xE) - 92*xE + sq(yE) + 2116));
float u2 = yE/sqrt((sq(xE) - 92*xE + sq(yE) + 2116));
float u3 = (xE + 23)/sqrt((sq(xE + 23) + sq((yE - 23*sqrt(3)))));
float u4 = (yE - 23*sqrt(3))/sqrt((sq(xE + 23) + sq((yE - 23*sqrt(3)))));
float u5 = (xE + 23)/sqrt((sq(xE + 23) + sq((yE + 23*sqrt(3)))));
float u6 = (yE + 23*sqrt(3))/sqrt((sq(xE + 23) + sq((yE + 23*sqrt(3)))));

float No [3][1] = {{No1},{No2},{No3}};
float u [3][2] = {{u1,u3},{u5,u2},{u4,u6}};
float uT [2][3] = {{u1,u3,u5},{u2,u4,u6}};

float fi = 10;

float T1 = fi*No1;
float T2 = fi*No2;
float T3 = fi*No3;

float Tnull[3][1] = {{T1},{T2},{T3}};


float DL1 = T1/Kl;
float DL2 = T2/Kl;
float DL3 = T3/Kl;

float deltaS1 = DL1*2000;
float deltaS2 = DL2*2000;
float deltaS3 = DL3*2000;


Serial.begin(38400);
Serial.println(deltaS1, DEC);
Serial.println(deltaS2, DEC);
Serial.println(deltaS3, DEC);


// Configure each stepper
stepper1.setMaxSpeed(200);   // step/second
stepper2.setMaxSpeed(200);
stepper3.setMaxSpeed(200);

// Then give them to MultiStepper to manage
steppers.addStepper(stepper1);
steppers.addStepper(stepper2);
steppers.addStepper(stepper3);

long positions[3]; // Array of desired stepper positions

positions[0] = deltaS1;
positions[1] = deltaS2;
positions[2] = deltaS3;
steppers.moveTo(positions);
```

```
 steppers.runSpeedToPosition(); // Blocks until all are in position
 delay(2000);



 }


void loop() {

 calculation_1(10,10);

 delay(1000);

 calculation_2(10,10);

 //reset(2.5,1.3,5);

Serial.end();
exit(0);


}
```

d) Appendix-III Codding of DC Motor

```
#include <Wire.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_ADXL345_U.h>


int in1 = 6;
int in2 = 7;
float motor_speed = 0;
int motor_control_pin = 6;

Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);


void setup() {


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_control_pin, OUTPUT);

pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);

digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);

}

void loop() {

  motor_speed = map(analogRead (A1), 0,1023,0,75);

  analogWrite(motor_control_pin,motor_speed);

  sensors_event_t event;
  accel.getEvent(&event);


  Serial.println(A1);
```

```
/* 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);



}
```