DEVELOPMENT OF STEM KITS FOR ENGINEERING STUDENTS

By

Ali Raza Khalid

Muhammad Shozab Khan

Awais Arshad

Ahamad Nawab

Department of Mechanical Engineering

School of Mechanical & Manufacturing Engineering

National University of Sciences & Technology

Islamabad, Pakistan

June, 2021

Development of STEM Kits for Engineering Students

A Final Year Project Report

Presented to

SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING

Department of Mechanical Engineering

NUST

ISLAMABAD, PAKISTAN

In Partial Fulfillment
of the Requirements for the Degree of
Bachelors of Mechanical Engineering

by

Ali Raza Khalid

Muhammad Shozab Khan

Awais Arshad

Ahmad Nawab

06/2021

EXAMINATION COMMITTEE

We hereby recommend that the final year project report prepared under our supervision by:

ALI RAZA KHALID	00000212518.
MUHAMMAD SHOZAB KHAN	00000207562.
AWAIS ARSHAD	00000210811.
AHMAD NAWAB	00000209388.

Titled: "**Development of STEM Kits for Engineering Students**" be accepted in partial fulfillment of the requirements for the award of BACHELOR OF MECHANICAL ENGINEERING degree with grade ___

Supervisor: Dr Rehan Zahid, Assistant Professor SMME	____________ Dated:
Committee Member: Abdul Naeem Khan, Assistant Professor SMME	____________ Dated:
Committee Member: Riaz Ahmad Khan, Assistant Professor SMME	____________ Dated:

______________________ ___________

(Head of Department) (Date)

COUNTERSIGNED

Dated: ________________ __________________________

(Dean / Principal

ABSTRACT

The aim of this project was to make STEM Kits that could be used by engineers for their learning and practical demonstration of theoretical concepts. These kits were made by blending the experimental apparatuses and STEM kits by combining their pros and overcoming their cons to provide a better learning experience. Salient features of these STEM kits are that they are multi-purpose, portable, modular, compact, provide us with digital measurements and have parts with flexible usage. We focused on several variations two kinds of experiments: 4-bar mechanism and gear train.

The method we used in this project was first to make an innovative design according to our needs and requirements with some preliminary calculations followed by animations, FEM simulations and advanced calculations while continuously improving the preliminary design through iterations. Some potential materials were identified and after careful consideration we selected best material for every component. Modern software tool used in this project include SolidWorks, Linkage, Mathcad and Excel. After the mechanical design was finalized, to have control on input motor so that we can vary the speed at which our primary shaft or input point rotates and to measure the RPM of various rotating shafts in gear train and 4-bar assemblies for which we designed electrical circuit and components for operation of our STEM KIT. All components including motors and sensors were selected after careful consideration. The electrical circuit is simulated in Proteus Design Suite.

The results of animations and simulations showed that the requirements were fulfilled. In animation, no hindrance was observed in motion. Also, in simulation, stress was higher than yield stress and no buckling was observed. Working of electrical circuit was also verified in Proteus.

ACKNOWLEDGMENTS

First of all, we would like to thank Allah Almighty, who is the most beneficent and merciful, for granting us with the wisdom and the ability to pursue and deliver this project. Without his blessings and help, we would not have been able to be where we are now. We would like to express our gratitude to our supervisor Dr. Rehan Zahid for his continued support and guidance. We completed this project under his patronage. We would like to thank him for believing in us and helping us develop as professionals. We would also like to acknowledge and give thanks to our committee members, AP Abdul Naeem Khan, Dr. Shamraiz and Dr. Riaz Ahmad for constant support and feedback. We also acknowledge the guidance and teachings of all SMME faculty members, especially those who taught us during our degree or who have been directly or indirectly involved in our professional learning and intellectual growth as engineers, making us able to complete this project. We would like to acknowledge all the authors of books and the media libraries which aided us in drafting this.

We would like to acknowledge our families for their unconditional support and love these last few years, without which we would not be here today. We would also like to acknowledge the support of our friends, of department as well as hostel for their help and support. Lastly, we would like to acknowledge and thank all the Healthcare professionals, Law enforcement officers and workers as well as scientists, engineers, delivery workers, daily wagers and frontline workers, working to combat the current COVID-19 pandemic, for their brave service and efforts.

ORIGINALITY REPORT

STEM KITS 2

ORIGINALITY REPORT

7%	5%	2%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	1%
2	Submitted to Engineers Australia Student Paper	1%
3	Submitted to Salalah College of Technology Student Paper	<1%
4	pt.slideshare.net Internet Source	<1%
5	b-ok.org Internet Source	<1%
6	erepository.uonbi.ac.ke:8080 Internet Source	<1%
7	www.freepatentsonline.com Internet Source	<1%
8	www.thread-rite.com Internet Source	<1%
9	Submitted to Sheffield Hallam University Student Paper	<1%

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

STEM	Science, Technology, Engineering, Mathematics
CAD	Computer Aided Design
DOF	Degree Of Freedom
FOS	Factor Of Safety
SFDs	Shear Force Diagrams
BMDs	Bending Moment Diagrams
RPM	Rotations Per Minute
PWM	Pulse Width Modulation
LED	Light Emitting Diode
LCD	Liquid Crystal Display
ABS	Acrylonitrile Butadiene Styrene
PET	Polyethylene terephthalate
PEEK	Polyetheretherketone
PP	Polypropylene

NOMENCLATURE

A_t	Tensile Stress Area
N	Number of teeth
m	Module of gear
α	Pressure angle
D	Pitch diameter
DO	Outer diameter
DR	Root diameter
DB	Base diameter
r_f	Fillet radius
Z	Number of teeth of bevel gear
δ	Pitch angle
E	Elastic modulus
I	Moment of inertia
M	Applied moment
F	Applied force

CHAPTER 1: INTRODUCTION

1.1 Motivation of work

STEM kits are a very common and great tool for learning subjects of Science, Technology, Engineering and Mathematics. The nature of these kits promotes practical learning and fosters students interests in these areas. Most of the STEM Kits available in market are just toys for children to help them understand very basic fundamentals of respective domains. They aren't suitable for learning at higher levels. Also, most of the experimental setups in engineering labs can perform only one kind of experiment and are usually very costly. If some advancements are made to STEM kits, so that they could perform Engineering experiments, we can overcome these problems.

1.2 Problem Statement

Design and development of an indigenous, portable, modular, learning and experimentation kit for students of science and engineering at college and university level. Kit should be designed to give students hands on experience and teach fundamentals of some important and popular engineering concepts.

1.3 Objectives

Our objective is to develop a STEM kit with the following features:

1.3.1 Modular:

Kit should consist of modular different parts assembled together and will have a bench on which different parts can be added to perform an experiment. These parts can be disassembled, assembled or rearranged anytime easily. This also leaves roam for any future additions to be made.

1.3.2 Multipurpose:

These modular STEM kits should be multipurpose and provide way for different variations or different kind of experiments to be performed on a single apparatus reducing complexity, cost and space.

1.3.3 Portable:

To increase the access and learning opportunities, the STEM kits must weigh as less as possible.

1.3.4 Compact:

These Kits should be compact and take as less space as possible to improve portability.

1.3.5 Flexible parts:

The parts used in STEM kits should be flexible in terms of usage. A single part may be used for different purposes in the apparatus or could incorporate change in dimensions e.g., a bar in 4-bar experiment should be able to be used for different required lengths in different variations.

1.3.6 Digital measurements:

STEM kits should provide digital measurements of input/outputs measured through suitable sensors and actuators.

1.3.7 Cost Effective:

The kit should be cost effective so it is affordable for more people and can be used for learning by a larger fraction of target audience.

1.4 Deliverables

Shortlisting potential experiments to be performed.

Plan Input and outputs of selected experiments

Develop a suitable modular and portable design based on potential experiments and their input/output.

CAD Modelling of parts and assembly of design in SolidWorks.

Run animations to confirm mobility of parts.

Shortlisting appropriate materials for parts and selecting the best one for each part individually.

Use conventional and modern software tools for analysis of design and make any required modifications.

Make an electrical design by selecting suitable sensors and actuators and program design.

CHAPTER 2: LITERATURE REVIEW

2.1 STEM

STEM (Science, Technology, Engineering and Mathematics), also called SMET (Science, Maths, Engineering and Technology) in past, is an expansive term to assemble these scholarly subjects in an applied and interdisciplinary force. This term is normally used to address a curriculum plan or educational strategy decisions in schools. It has indications for labor force improvement, workplace automation, immigration policy and public safety concerns.

2.2 STEM Kits

STEM kits are a common and great tool for learning subjects of Science, Technology, Engineering and Mathematics. The nature of these kits promotes practical learning and fosters students interests in these areas. Many STEM kits tend to target one or two of subjects mentioned beforehand, but there is a lot of overlap between them.

In market, many examples of such kits and some classic toys are available for teaching children in an educational and fun way. Some examples of timeless STEM kits you can buy are 'Old school' toys like Mechano or LEGO. Under the banner of STEM, these kits engage students in hand-on experience and imparts practical learning.

2.2.1 Features of STEM Kits

Salient features of STEM kits are mentioned below:

- Portable
- Modular
- Multipurpose
- Cost-effective

Figure 1: Some examples of STEM kits.

2.2.2 Benefits

Following are some of the benefits of STEM Kits:

- Boost curiosity,
- Enhance creativity,
- Improve cognitive skills,
- Develop hand-eye coordination capabilities,
- Empower critical skills,
- Fun and Interactive,
- Enhance divergent thinking,
- Improve problem solving skills.

2.3 Experimental Apparatus in Engineering Labs

The experimental apparatus present in engineering labs provides us with a platform for advanced practical learning as STEM kits cannot incorporate such complex and higher-level experiments particularly for Mechanical Engineering.

2.3.1 Features of Engineering Experimental Apparatus:

Salient features of experimental apparatus are mentioned below:

They offer practical learning for complex theoretical tests.

These apparatuses usually take up a lot of area.

They are not modular in most cases.

Existing experimental kits can perform experiments of one single kind.

Most of the times, they do not cater to different variations of same type of experiment (e.g., in case of 4 bar mechanisms)

Generally, they are very expensive.

Analog measurement methods are used in most of experimental kits and digital equipment are costs much more.

Figure 2: Basic Gear Train Apparatus

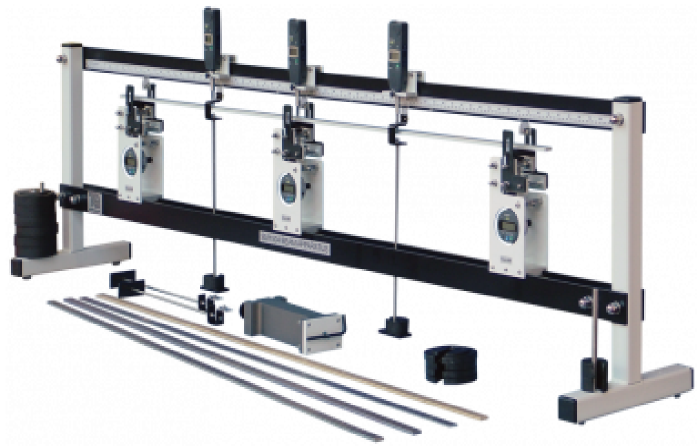


Figure 3: Deflection of beam apparatus

Figure 4: 4-bar (Crank-rocker) apparatus

Figure 5: 4 bar (Slider crank) apparatus

2.4 STEM Kits for Engineering Student

STEM Kits for engineering students, that we will develop, will be blending the experimental apparatuses and STEM kits by combining their pros and overcoming their cons to provide a better learning experience.

2.4.1 Need for STEM Kits for Engineering Students:

We need these improved STEM kits due to the following reasons.

Existing experimental kits can perform only limited set of experiments on a single apparatus.

Buying different apparatus for different kind of experiment is usually very costly.

Most experimental apparatuses do not incorporate digital measurements and analog method of measurement is used.

Most lab equipment used in Pakistan is made by foreign companies (Tecquipment, UK; Edibon, Spain; Hamburg, Germany; Armfield, USA etc.)

Foreign made STEM Kits are mostly for kids ranging from $60 to $500 (Engino, Stem-supplies, Demco, etc)

STEM KITS made in Pakistan are very elementary, containing some LEDs, motors, magnets with no substantial learning for engineering students. (Sciencestore.pk, Circuit.pk, Popuplearning.pk)

2.5 List of experiments

The main purpose of these STEM kits for engineering students is to perform different kinds of experiments along with their multiple variations. These kits are able to perform different variations of gear train experiments, deflection of beam experiment, experiment for calculating friction between surfaces, experiments for different variations of 4-bar mechanism, and many other experiments. In the scope of this subject, we will go in detail of the following two experiments in detail:

Gear Train experiment (with different variations)

4-bar Mechanism experiment (with different variations)

2.5.1: Gear Train Experiment:

In this experiment we consider different variations and mechanisms of a gear assembly. The kit being modular provide means to disassemble and assemble to try and run a different mechanism alone or along with another mechanism. The main functions of gear drive include:

To transmit power from one point in assembly to another.

To change direction of rotating shafts.

To increase/decrease torque to driven equipment from driving equipment (motor).

To increase/decrease rpm generated by motor.

The shaft arrangement on gear drive is the relative location to each other of high-speed and low-speed shaft extensions. This is directed by the position of the equipment driven and of the motor and sometimes constrained due to physical space of the application. The two types of gears used in out kit are:

Spur Gear:

Spur gears has teeth parallel to the shaft axis and transmit power through shafts that are parallel. Due to this, gears produce radial reaction loads on the shaft rather than axial loads. These gears only have one tooth in contact with other gear at a time. The teeth release contact with a tooth and accelerate to get in contact with the next tooth while they are rolling through mesh.

Bevel gear:

Bevel gears are used when there is a need of right-angle gear drive. The transmit power between shafts that are intersected at an angle of 90º. They do not generally transmit as high torque (per size) as parallel shaft arrangement.

When power is held constant, torque and speed are proportional and inversely related to each other. Therefore, as the speed increases, torque decreases and as the speed decreases, torque increases. Some mechanisms that are used in gear train experiment are explained below.

Figure 6: Spur and Bevel gear

2.5.1.1 Simple Gear Train:

When there is only one gear on each shaft, it is known as simple gear train. In a simple gear train, the driving and driven shaft (or a low speed and high-speed shaft) are parallel to each other on the same horizontal shaft. This type of gear drive type is used when there is a need of transmitting a high torque. Transmission can be done through either of helical gear, spur gear or internal gear. In our case, we have used spur gear for parallel drive transmission in model.

In figure 9A, gear 1 & 2 transmit motion from one shaft to another. 1 is the driving gear and 2 is called follower or driven gear. As we know that rpms of driven gear will be different from driving gear and that no. of teeth and rpm is proportional and inversely related to each other in a simple gear train. So,

$$Speed\ Ratio = \frac{N_1}{N_2} = \frac{T_2}{T_1}$$

where,

$\mathrm{N_1} = \text{Speed of gear 1 in rpm}$

$\mathrm{N_2} = \text{Speed of gear 2 in rpm}$

$\mathrm{T_1} = \text{No. of teeth of gear 1}$

$\mathrm{T_2} = \text{No. of teeth of gear 2}$

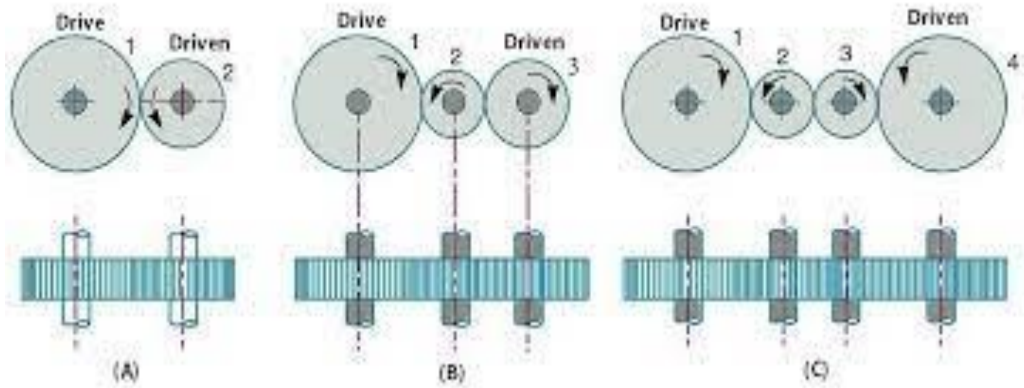


Figure 7: Different variations of simple gear train.

In figure 9B, 1 is the driving gear, 3 is called follower or driven gear, while 2 is intermediate gear. When driving gear 1 is in mesh with intermediate gear 2, speed ratio is given as:

$$Speed\ Ratio = \frac{N_1}{N_2} = \frac{T_2}{T_1}$$

When intermediate gear 2 is in mesh with driven gear 3, speed ratio is given as:

$$Speed\ Ratio = \frac{N_2}{N_3} = \frac{T_3}{T_2}$$

By multiplying the above two equations, we can get speed ratio of gear train.

$$\frac{N_1}{N_2} \times \frac{N_2}{N_3} = \frac{T_2}{T_1} \times \frac{T_3}{T_2}$$

We get,

$$\frac{N_1}{N_3} = \frac{T_3}{T_1}$$

Hence,

$$\text{Speed Ratio} = \frac{r.p.m. of\ driver}{r.p.m. of\ driven} = \frac{No. of\ teeth\ on\ driven}{No. of\ teeth\ of\ driver}$$

Similarly, in figure (C), gears 2 and 3 are intermediate gears and speed ratio is:

$$\frac{N_1}{N_4} = \frac{T_4}{T_1}$$

Intermediate gears do not affect speed ratio and due to this reason, they are also known as idle gears. They serve the following purposes: To get desired direction of motion (i.e., clockwise, or counterclockwise) on follower gear, and to connect gears at large center difference.

2.5.1.2 Compound Gear Trains:

When there are more than one gears on a shaft, it is called a compound gear train. Single gearset of spur, helical or bevel gears usually limited to ratio of about 10:1. Gearset become very large, expensive, and hard to package above that ratio if pinion kept above minimum numbers of teeth. Compound gear trains allow us to multiply gear trains together. Thus, making us capable of generating very large torques and very low speeds from low torque, high speed motors by generating very large gear ratios.

The overall gear ratio is found by multiplying all the individual gear ratios together.

$$\frac{N_1}{N_6} = \frac{T_2}{T_1} \times \frac{T_4}{T_3} \times \frac{T_6}{T_5}$$

$$\text{Speed Ratio} = \frac{r.p.m.of\ first\ driver}{r.p.m.of\ last\ follower} = \frac{Product\ of\ no.of\ teeth\ of\ driven}{Product\ of\ no.of\ teeth\ of\ drivers}$$

2.5.1.2 Right angle gear drive:

Right angle gear drive is used when driving and driven shafts are perpendicular to each other, and torque or rpm is required to be transmitted at an angle of 90º. Applications of these drives include conveyor belts, drive train of cars or other applications, in which driving equipment to be close to the driven equipment.

2.5.2 4-bar Experiment:

In this 4-bar experiment we consider different variations of a 4-bar assembly. The kit being modular provide means to disassemble and assemble to try and run a different variation.

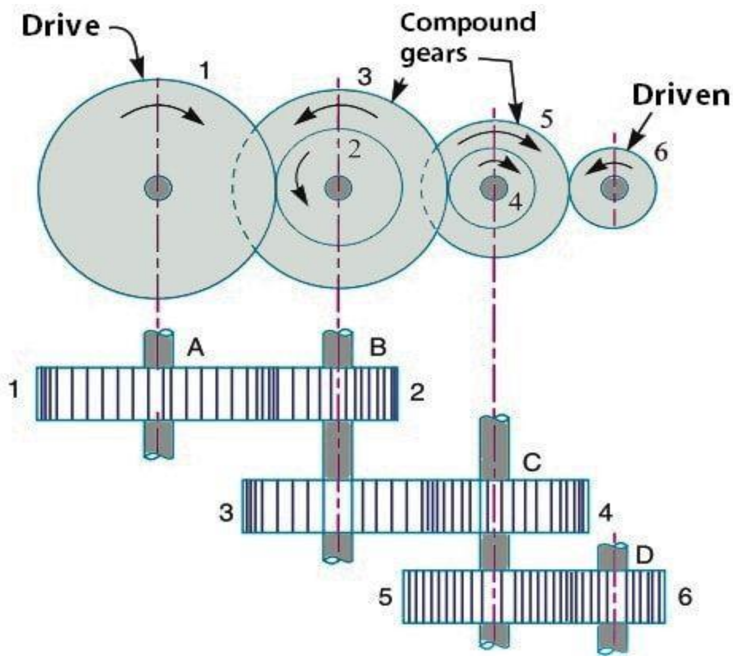


Figure 8: Compound gear Train.

Figure 9: Right angle gear drive.

A planar four-bar linkage consists of four rigid rods connected in a plane through pin joints. It has N_{DOF} = 1 degree of freedom. For the following mechanical purposes 4-bar mechanism can be used:

Convert rotational motion to reciprocating motion.

Convert reciprocating motion to rotational motion.

To magnify force.

To constrain motion.

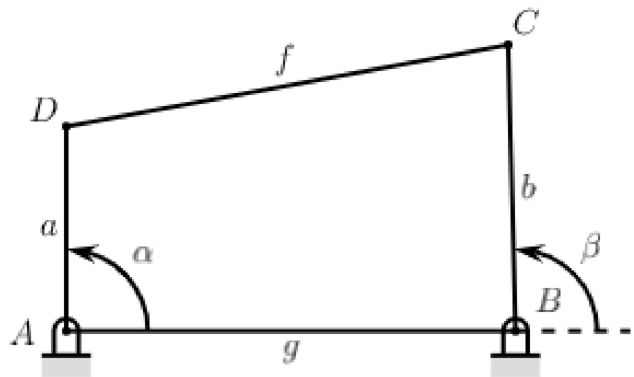


Figure 10: 4 bar Mechanism.

Here,

a is input link (crank)

f is coupler

b is output link (rocker)

g if the ground link

Input link rotates full only when Grashoff condition is satisfied. Grashoff condition is satisfied when:

$S+L<P+Q$

Where,

S=Smallest link among the four links.

L=Largest link among all links.

P=1st remaining link

Q=2nd remaining link

Following are the different variations of 4-bar performed in STEM kits:

2.5.2.1 Crank-Rocker configuration:

In the crank rocker configuration, the rocker oscillates between two limiting angles, while the crank takes full rotation. It has applications in wind-screen wiper etc.

2.5.2.2 Crank Slider configuration:

In this 4-bar configuration one link slider over the ground link while crank takes full rotation. It has applications in engine piston etc.

CHAPTER 3: METHODOLOGY

Our team conducted this project in 5 main phases that are as follows:

Mechanical Design

Material Selection

Design Validation or Analysis

Electrical Design

3.1 Mechanical Design

The design phase was initiated in CAD Modelling for Base, side-carriage beams, support beams, gears, meshing profile, Geneva mechanism, bevel gear and motor foundation. The design was initiated keeping in view the maximum breadth and height of STEM kit and lengths if individual parts were initially conjectured so that they were iteratively reduced to minimum values afterwards to prevent deformation. A safety factor (FOS) of 4 was assumed throughout the design.

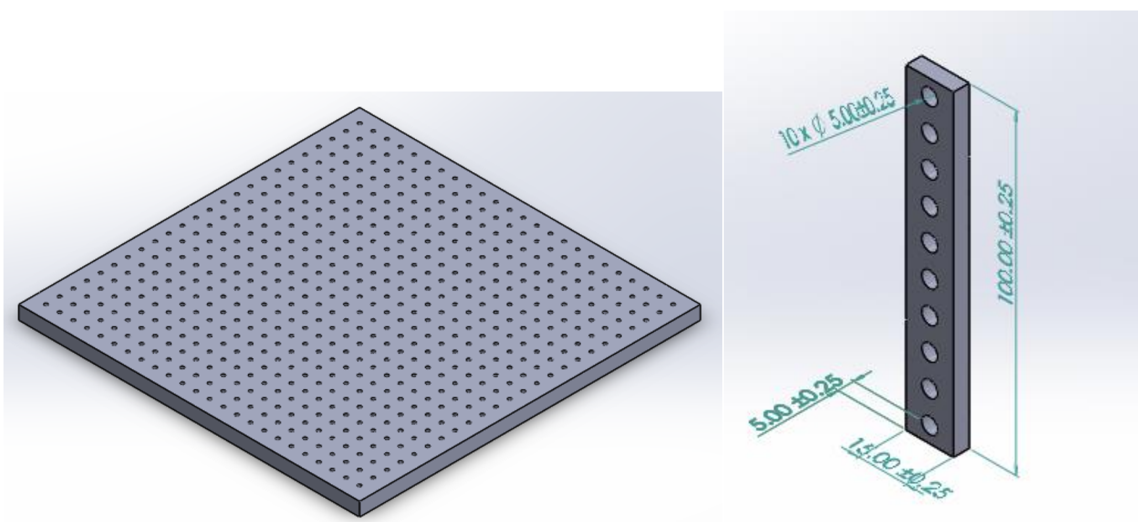


Figure 11: Base and Support beam.

3.1.1 Base Design

A common base was made with a 50cm X 50cm area, the thickness was arbitrarily selected as 1 cm but after doing analysis on support beam, it was undergoing fracture, it was changed to 20cm.

3.1.2 Side Carriage Beams

These beams house the sliders for executing 4-bar crank-slider motion. Moreover, these side carriages also provide primary support to gear train and is subjected to torsion and flexural bending from stresses generated by the meshed-up gear train. With 290mm in length added with 10mm distance from both sides, it forms a 300mm carriage. Four, 5mm holes provide pin support for the carriage beam with the support beam.

3.1.3 Support Beam

The support column supports the side carriage and has linearly patterned holes to adjust height of the carriage. Moreover, these holes add to the versatility of the beam used as support for different configurations.

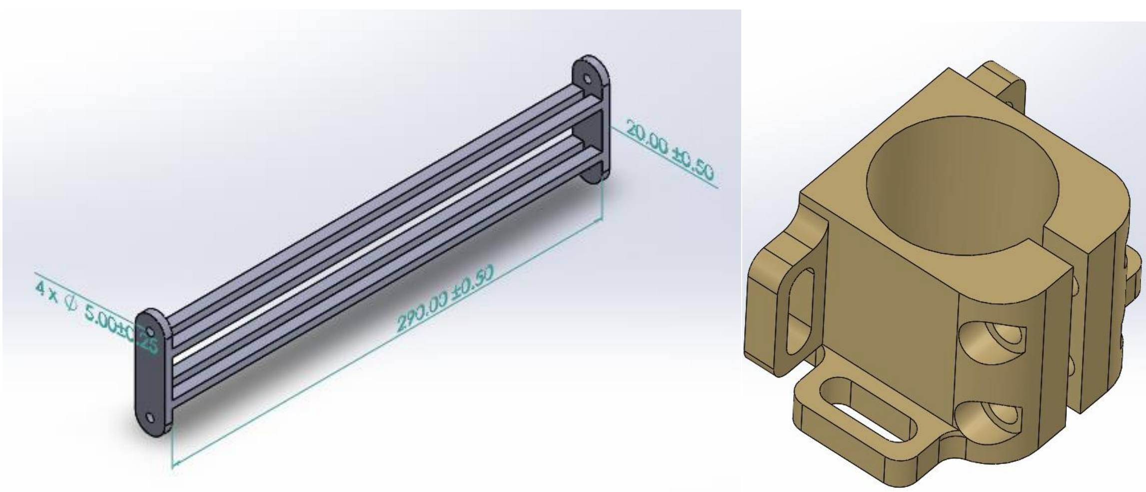


Figure 12: Side Carriage beam and Motor support

3.1.4 Motor Bracket

A motor is to be used to drive gears as well as to drive the four bar mechanisms. The motor is supported by a motor bracket which allows a bolt to be inserted from a 3mm hole and tightened on the other side by a hex nut. This mechanism allows large tolerances for motor diameter and is finally locked by a HEX-3 nut and a bolt.

3.1.5 Spur Gears

For designing spur gears standard used is “Involute Full-depth Teeth ANSI Course Pitch Spur Gear Teeth Forms ANSI B6.1”. The parameters and the formulas used are stated in the table below.

Table 1: Spur gear parameters.

Parameters	Formulas	Values
Module (m)	-	2 mm
Number of teeth (N)	-	16
Pressure angle (α)	-	20°
Addendum (a)	$1.0 * m$	2.0 mm
Dedendum (b)	$(1.0 + c) * m$	2.5 mm
Clearance (c)	$0.25 * m$	0.5 mm
Pitch diameter (D)	$N * m$	32.0 mm
Outside diameter (DO)	$(N + a) * m$	36.0 mm
Root diameter (DR)	$(N - b) * m$	27.0 mm
Base diameter (DB)	$D * \cos \alpha$	30.07 mm
Fillet Radius (rf)	$0.3 * m$	0.60 mm

Backlash	$0.04 * m$	0.08 mm

Furthermore, the screenshot showing formulas defined and used for dimensions while designing spur gear in SOLIDWORKS is attached in Appendix I.

After defining the parameters and calculating some for the importance variables, we started with drawing pitch circle, outside circle, and root circle on a sketch. To draw base circle, a line (1) tangent to pitch circle was drawn horizontally (point X), and a line (2) originating from same point at an angle (equal to pressure angle) was drawn; then the circle was drawn tangent to that second line. A line (3) connecting base circle and outside circle passing through point X was drawn and slit in half. Then, a line (4) from mid-point (point Y) from of previous line (3), and tangent to base circle (point Z) was drawn. Line (4) is extended beyond tangent point so that ratio of its length to total length is 1:4. An arc was made centered at point Z and passing through point Y, extending from root circle to outside circles. A line (5) is drawn from origin to outside circle at half-tooth angle from line (3). Then, the arc drawn was mirrored about line (5). This completed the tooth profile of spur gear.

Root circle and tooth profile was extruded 15mm, producing the spur gear. Fillet and backlash were added according to the formulas mentioned above. Holes were added for mounting the gear to shaft using gear coupling: 5mm hole in center of shaft and 3mm holes around. Figures below shows sketches used made and extruded gear. By changing the value of variable “N”, we produced more gears with even number of teeth i.e., 16, 18,20, … .

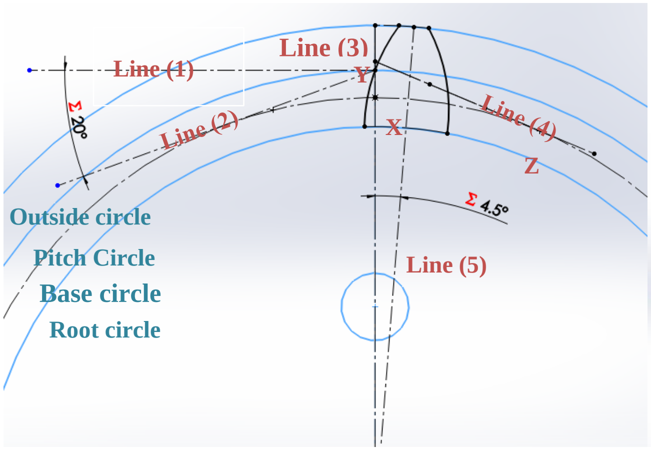


Figure 13: Spur gear teeth geometry.

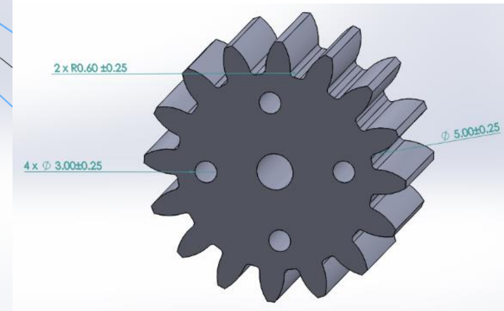


Figure 14: Extruded spur gear.

3.1.6 Bevel gears

For designing bevel gears the parameters and the values selected are stated in the table below:

Table 2: Bevel gear parameters.

Parameters	Formulas	Values
Module (m)	-	3 mm
Pressure angle (α)	-	20°
Shaft Angle (Σ)	-	90°
Number of teeth (Z_1, Z_2)	-	17, 17
Pitch Diameter (D_1, D_2)	$m * Z$	51 mm, 51 mm
Pitch angle (δ)	$\delta_1 = \tan^{-1}(\frac{\sin \Sigma}{z_2/z_1 + \cos \Sigma})$	45°

	$\delta_2 = \Sigma - \delta_1$	45°
Cone distance (R_e)	$\frac{d_2}{2 * \sin \delta_2}$	36.06
Face width (L)	Less than m*10 OR R_e /3	15 mm
Addendum angle (θ_a)	$\tan^{-1}(2 * \sin \delta / z)$	4.755°
Dedendum angle (θ_f)	$\tan^{-1}(1.25 * 2 * \sin \delta / z)$	5.936°
Virtual pitch diameter	$\frac{d}{\cos \delta}$	72.12°
Virtual outside diameter	$\frac{d + 2 * m * \cos \delta}{\cos \delta}$	78.12°
Virtual root diameter	$\frac{d - 2.5 * m * \cos \delta}{\cos \delta}$	64.62°
Virtual base diameter	$\frac{d * \cos \alpha}{\cos \delta}$	67.77°
Virtual root dia undercut	(Virtual base diameter x 0.99)	64.62 mm

For modelling bevel gear, a sketch was made defining geometry using pitch diameter, pitch angle, face width, addendum distance, dedendum distance, a small distance (dependent on pitch diameter) was added below dedendum distance to provide support area below tooth. After revolving the previous sketch, another sketch was made on outside face of teeth. On this sketch we made all four virtual diameters and my marking half tooth angle we drew the cut tooth profile and mirrored it.

Cut tooth profile was swept towards the apex, and this cut was patterned around the diameter to produce successive cuts and corresponding teeth. A 14mm collar was made on the bottom, through which 5mm hole passes and a 3mm transverse hole for mounting on shaft. After completing the gear, two bevel gears of 17 teeth each were assembled together using a support arm.

Furthermore, the screenshot showing formulas defined and used for dimensions while designing bevel gear in SOLIDWORKS is attached in Appendix II.

3.1.7 Geneva Mechanism

A Geneva mechanism is incorporated in the kit to introduce the students with basic mechanism to perform an intermittent rotary motion. Conversely the experiment can be carried on by forming a Geneva train to calculate speed of the motor by observing the number of slots surpassed by final star-drive.

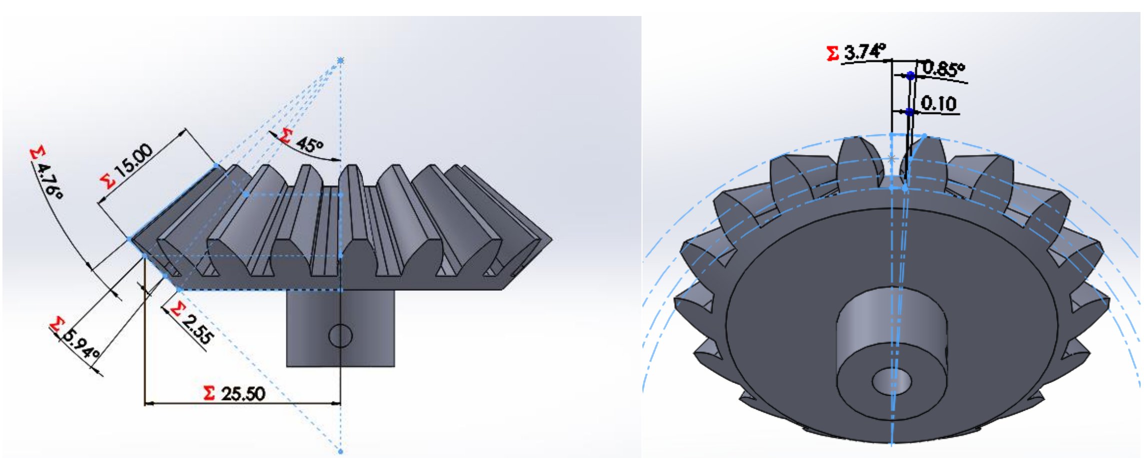


Figure 15: Bevel gear sketches used.

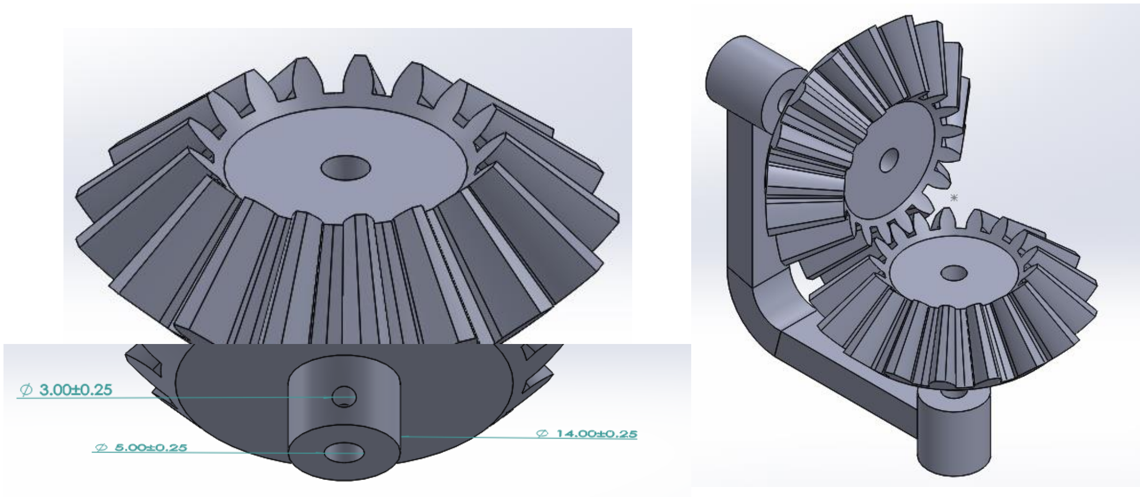


Figure 16: Finished and assembled bevel gears.

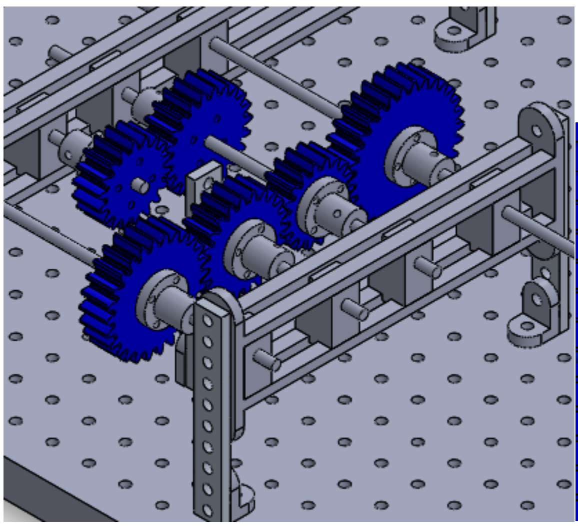


Figure 17: Assembled gear train.

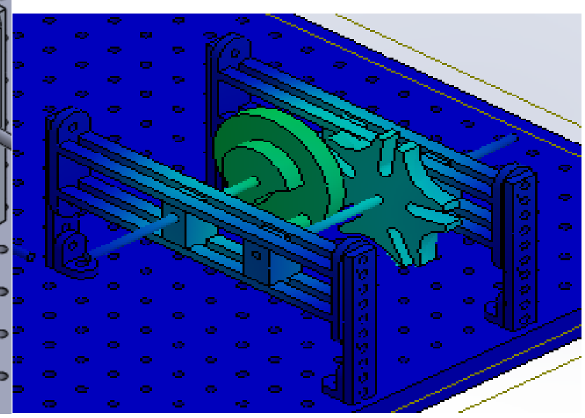


Figure 18: Geneva mechanism.

3.2 Material Selection

For a STEM kit to be compact, lightweight yet endurable and robust enough, material selection is a crucial part of design. While selecting materials, we went through mechanical properties of commercial materials such as cast iron, steels, aluminum, to name a few. These materials had good yield strengths and fracture toughness, but their density was quite more than required and it was making the components heavy.

In order to keep STEM kit low-weight and at the same time keep their stress concentrations less than maximum stresses that the material can endure, we underwent rummage through readily available 3D materials. A lot many materials were available on internet with their usages contingent to operations they have to perform. For example, stain-less steels could be deployed as carriage rods with low coefficient of friction when lubricated. Similarly, cast iron could be use for bar mounts and gears but since they have to be deployed on a STEM kit, they cannot be used – the stem kit would get bulky otherwise.

Coming towards materials which are suited to 3D printing, we selected:

Acrylonitrile Butadiene Styrene (ABS)

Polyethylene terephthalate (PET),

Polyetheretherketone (PEEK),

Nylon 10/6,

Polypropylene (PP).

All of these materials are thermo plastic materials except nylon 10/6 which is a thermo poly amide. Among all, PEEK has highest tensile strength of 95 MPa, but PET and Nylon also have comparable strengths of 75 MPa. PP has lowest value of 28 MPa, and ABS is no exception in this regard.

We selected a criterion to select materials so they could be arranged in order to of their increasing match with desired properties. In order of priority levels, yield strength was to define ductility, fracture toughness for shock resistance, tensile strength to avoid any necking when region crosses its strain hardening or Ultimate Tensile Strength, to be more direct. At last, since manufacturing was not to be done, we selected machinability as a criterion for selection of material. Had there been manufacturing aspect involved, we would have set machinability in highest orders of selection. The manufacturing plan, nonetheless, is still a part of our deliverable.

Yield strength is of greater interest since gears meshed in gear trains and beams supporting them are more prone to yield than fracture so yield strength was given a priority over facture toughness. PEEK had yield strength of 75 MPa and nylon read 65MPa. The rest- ABS, PET and PP- show yield strength as low as 40MPa.

An Excel Sheet was formed which included yield strength, tensile strength, Elongation, workable temperature, and impact resistance to check for material's propensity to be selected for a part. The scores from 0-4 were given for very low to very high thru low, medium, and high in between.

The elastic modulus of all materials is less compared to metals but sufficient enough to withstand scaled-low operations of our STEM kits. PEEK has elastic modulus of 3400 and nylon reads 2950. Comparing elastic modulus with yield strength, we selected *nylon* for gears and PEEK for side-carriage beams. The reason being nylon has better yield strength and comparable fracture toughness. PEEK was selected for side-carriage beams because of its better torsion resistance.

For non-critical parts i.e., those who are not subjected to dynamic loading, we select nylon due to its readily availability and economical price.

A summary for materials selected for critical components of our assembly are given below:

Table 3: Material for each component.

Assembly Component	**Material**
Gears	Nylon
Couplings	Nylon
Side Carriage beams	PEEK
Slider blocks	Nylon
Support members	PEEK
Motor support	PEEK
Shafts	PEEK
Support brackets	Alloy 1060
Riveted nuts	Stainless Steel

Cap screw Allen bolts	Stainless steel

3.3 Structural Analysis

3.3.1 Static Analysis

First of all, static analysis was done to ensure no-deformation of our STEM kit. The mass of gears was assumed to be divided equally on side-carriage beams at contact points and with these point loads, static analysis of side-carriage was carried out.

The gravitation analysis for whole kit was done in SolidWorks to ensure no part deforms in static standing position. The results were encouraging as the maximum deformation was observed in input gear which was of miniscule magnitude of 0.082mm.

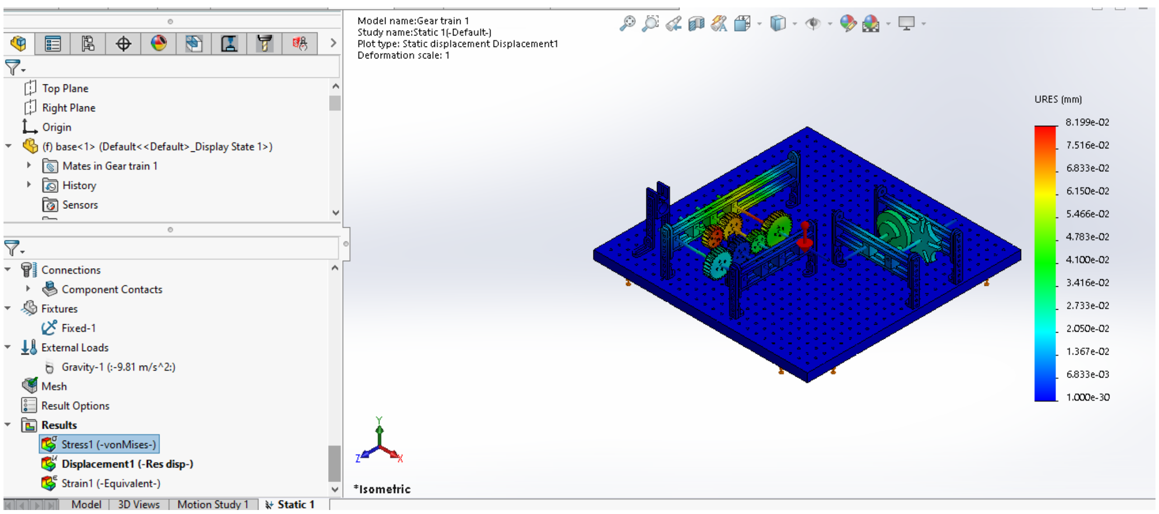


Figure 19: Full assembly gravitational analysis.

Moving onwards to analysis of critical parts, we selected side-carriage beams and column supports as our critical parts by intuition. The two side-carriage beams are both of different lengths but house similar stresses from gear trains and their housing shafts. With a 1.09N force concentrated at 7.5mm, 55.5mm, 93.5mm and 143.5mm from x=0 of the beam, the concentrated forces were analyzed for Shear Force Diagrams (SFDs) and Bending Moment Diagrams (BMDs) to check for pure bending and deformation resulting from stresses. The BMDs and SFDs are drawn on MathCAD and the results were compared with those generated with SolidWorks and were seen in agreement with an error of 7% only. The maximum Von Misces stresses were observed to be only 1.045MPa compared to yield stress of PEEK at 75 MPa (Figure. 20).

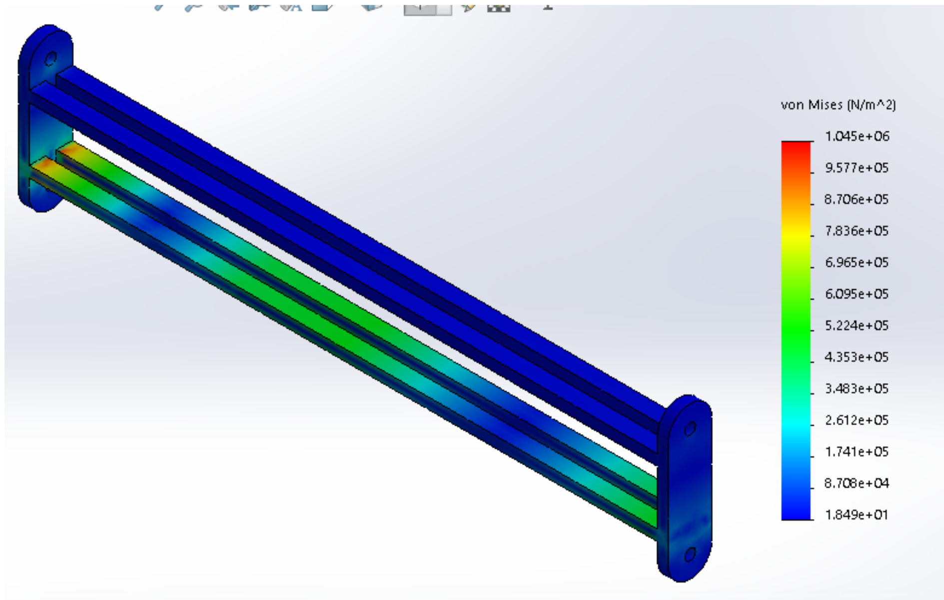


Figure 20: VonMises stress on Side carriage beam.

The maximum deformation is observed from 55.5mm <x< 93.5mm in our calculations but is of as low as 0.24mm which is justified (Figure. 21).

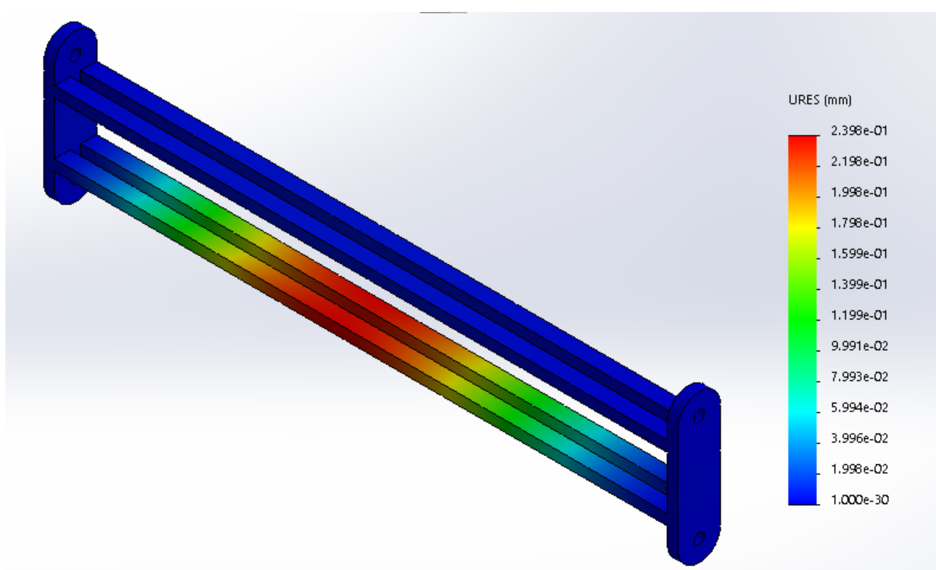


Figure 21: Deformations in Side carriage beam.

The small side-carriage beam also showed a similar pattern of maximum stresses and displacement, but the peak stresses was even lesser at 0.87MPa, 16% lesser than that experience by larger carriage beam. This is justified individually as smaller beam will experience lesser displacement having higher area moment of inertia as y'' = M/EI.

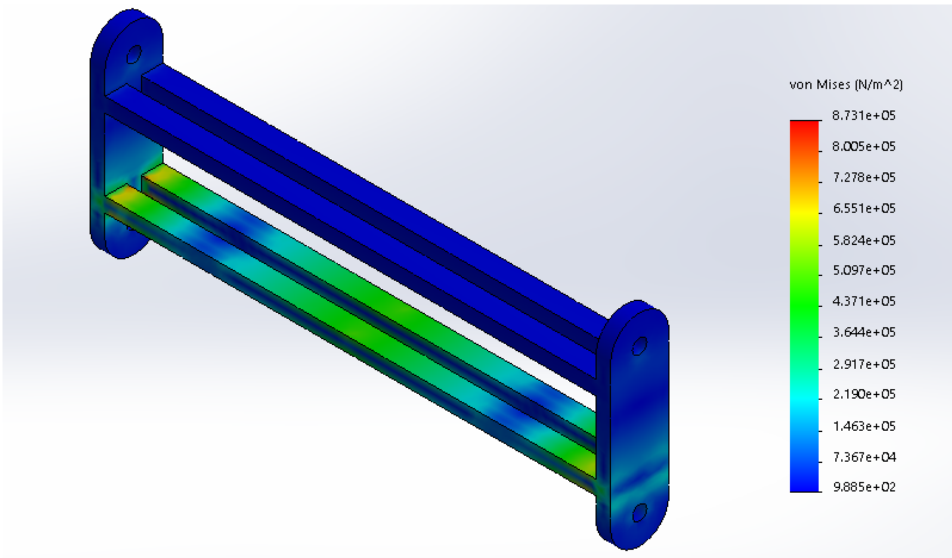


Figure 22: VonMises stress on small Side carriage beam.

The side column was also a critical part as it undergoes transverse buckling about x-axis in our coordinate system. The nodes of buckling were determined by SolidWorks analysis, but the buckling analysis was performed using MathCAD using Euler's criterion (Figure. 23). The calculation procedure was iterated for a metal and a plastic but compared to the compressive force borne by the column, the critical force and corresponds stresses were very large even after applying a factor of safety of 4. Nylon column gives a critical force of 2.6kN whereas our design produces compressive force of 1.7N. so it can infer as an overdesigned side column. For 1060 Alloy, the value of critical stress whoops up to 21.72kN.

Buckling of Side Support Column

Since the column is fixed from both ends, the equivalent length would be: **0.5*L**

For Nylon columns:

$E := 8.3\ GPa$

$b := 0.015\ m$

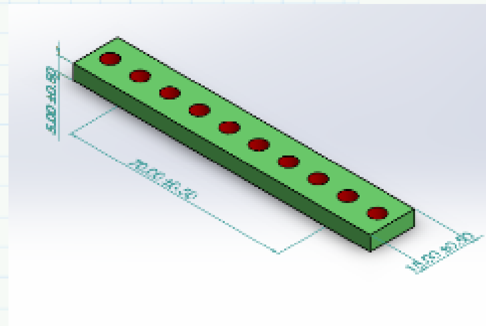
$a := 0.005\ m$

$L_a := 0.07\ m$

$I_x := \frac{1}{12} \cdot b \cdot a^3$

$L_e := 0.5 \cdot L_a$

$P_{cr'} := \frac{\pi^2 \cdot E \cdot I_x}{L_e^2} = (1.045 \cdot 10^4)\ N$



For a FoS of 4,

$P_{cr} := \frac{P_{cr'}}{4} = 2.612\ kN$

The Area of Cross section of column is:

$A := b \cdot a$

The critical buckling stress will be then,

$\sigma_{cr} := \frac{P_{cr}}{A} = 34.829\ MPa$

For **1060 Alloy** columns:

$E := 69\ GPa$

Figure 23: Buckling analysis of Side support column.

3.3.2 Analysis of Crank Slider Mechanism

Most of the stress in crank slider mechanism is due to the frictional effect of the sliding mechanism and vibration of the motor which powers the system. The friction of the sliding mechanism is mainly due to two forces:

1. The weight of the slider
2. The normal components of the acceleration on the slider which is due to the motion of the entire mechanism.

We integrated our angular acceleration for a whole interval of 0 to 360 degrees using Excel. The maximum angular acceleration using analytical formulas was found to be 57.3 rad/sec^2.

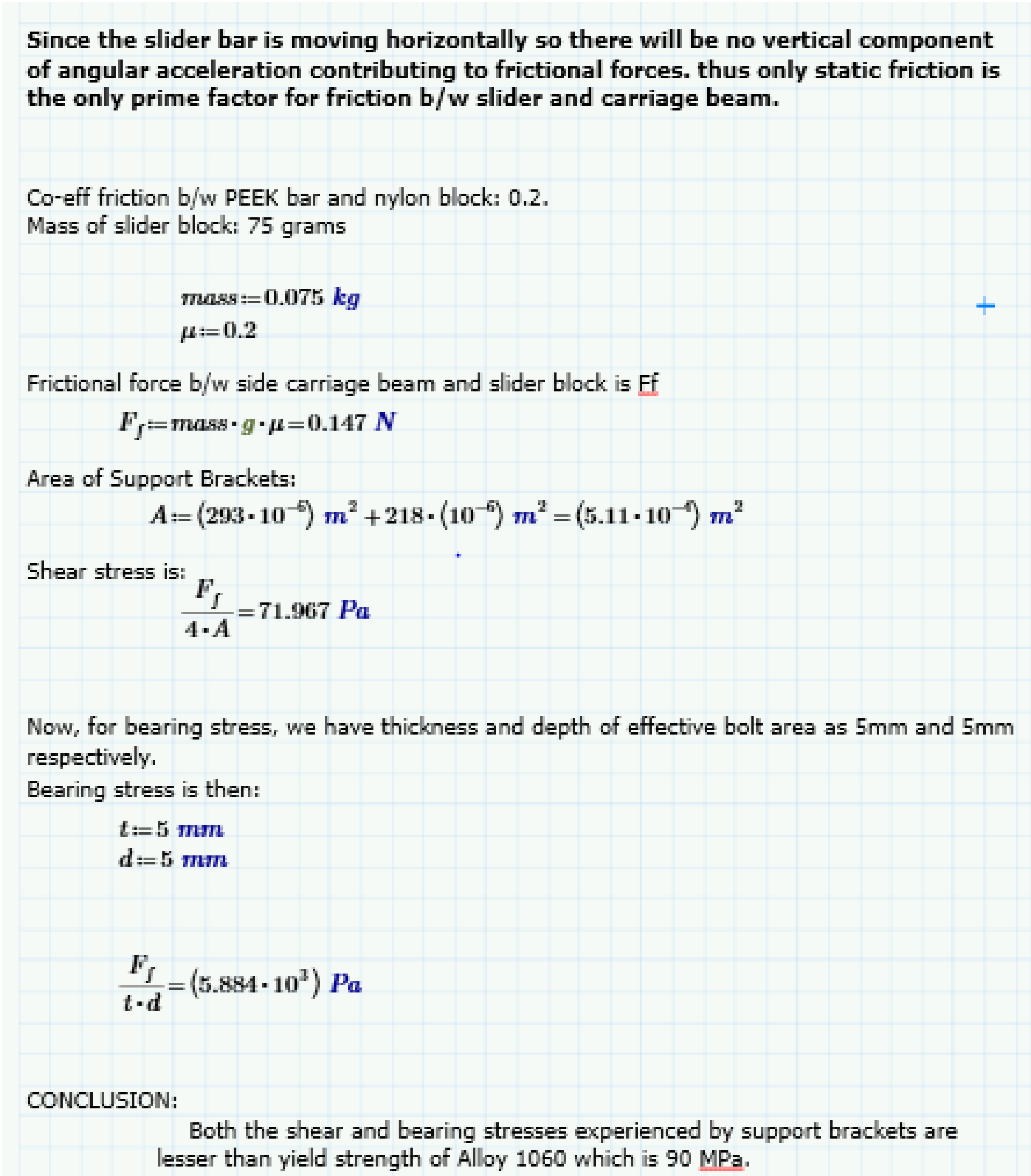
Our slider is moving only in horizontal direction, so no component of our tangential acceleration is contributing to frictional forces and hence only gravitational component is contributing to shear and bearing stresses in support brackets.

The friction due to the motion of the slider causes a force to be exerted in the entire mechanism which causes shear stress in the bolting attachments which can result in detachment of the arms from the base of the mechanism. The MathCAD file was made for calculating frictional forces that will result in transverse shear (in and out of plane) due to slider movements. The shear and bearing stresses were found to be 71.96Pa and 588.4 kPa. Both these values are quite low than the yield stress of our Alloy 1060 support brackets. We could not use 3D printable plastics for manufacturing of support brackets because they had to be gripped using stainless steel riveted nuts and bolts. Had we used nylon for support bracket, its threading would have worn out in matter of a couple of clamping and loosening. Screenshot of MathCAD file is attached below (Figure. 24).

3.4 Nut Selection

After comparing with a number of available nut and bolt combinations, we opted for riveted nuts in combination with hex bolts. The riveted nuts are most suited to our requirement of instant gripping and setting parts apart. Due to softer parts being selected for our assembly, we selected *CAK Small Flange Knurled Body Thin Wall series*. The catalogue mentions CAK series optimized for near flush requirements which matched with our own. Similarly, knurled body provides extra resistance to spin out when installed in softer materials primarily nylon and PEEK, in our case.

CAK-680-3.3 stainless steel riveted net with closed end is selected after reviewing literature for riveted nuts. It has a length of 12.07mm and grip range of 0.5 to 3.3 mm. hole size is 6mm which is in clearance fir with our assembly.



Since the slider bar is moving horizontally so there will be no vertical component of angular acceleration contributing to frictional forces. thus only static friction is the only prime factor for friction b/w slider and carriage beam.

Co-eff friction b/w PEEK bar and nylon block: 0.2.
Mass of slider block: 75 grams

$$mass := 0.075\ kg$$
$$\mu := 0.2$$

Frictional force b/w side carriage beam and slider block is Ff

$$F_f := mass \cdot g \cdot \mu = 0.147\ N$$

Area of Support Brackets:

$$A := \left(293 \cdot 10^{-6}\right)\ m^2 + 218 \cdot \left(10^{-6}\right)\ m^2 = \left(5.11 \cdot 10^{-4}\right)\ m^2$$

Shear stress is:

$$\frac{F_f}{4 \cdot A} = 71.967\ Pa$$

Now, for bearing stress, we have thickness and depth of effective bolt area as 5mm and 5mm respectively.
Bearing stress is then:

$$t := 5\ mm$$
$$d := 5\ mm$$

$$\frac{F_f}{t \cdot d} = \left(5.884 \cdot 10^{3}\right)\ Pa$$

CONCLUSION:

Both the shear and bearing stresses experienced by support brackets are lesser than yield strength of Alloy 1060 which is 90 MPa.

Figure 24: Stress analysis for 4-bar configuration in MathCad.

An *allen bolt* of stainless steel with socket cap hex head M5 of pitch 0.8 mm in coarse pitch series with tensile stress area A_t of 14.2 mm^2 and length 13 mm would be the best fit for this nut and can be used to create a strong yet configurable fit for our assembly parts.

Figure 25: Riveted Nut.

Figure 26: Cap screw Allen bolt.

3.5 Electrical Design

After completing mechanical design, we needed electrical circuit and components for operation of our STEM KIT. Firstly, we needed to provide rotatory power to shafts in gear train and 4-bar assembly, we needed to have control on input motor so that we can vary the speed at which our primary shaft or input point rotates. Secondly, we needed to measure the RPM of various rotating shafts in gear train and 4-bar assemblies. This control of input and measurement of output will provide opportunity to user to learn and explore the behavior of their desired mechanism.

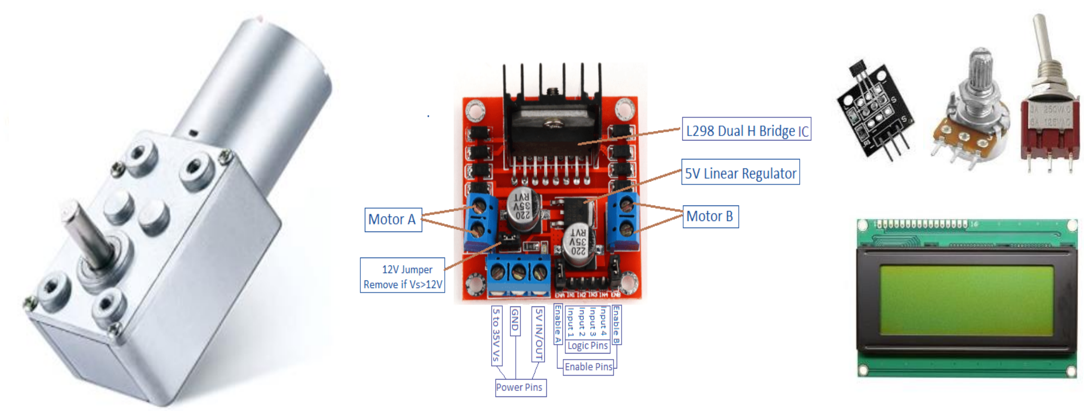


Figure 27: Components for electrical circuit.

Gear motor is used because the experimental nature of our equipment demands that shafts rotate with low speed or RPM and higher torque. In this way users can easily observe and learn from the rotation of gears or 4-bars. Table 4 contains detailed specifications of gear motor and motor driver proposed to be used.

Table 4: Specification of electrical components.

Gear Motor Specifications	
Model	JSX180-370
Rated Voltage	12V (DC)
Rated Current	0.6A
No-Load Speed	50 RPM
D Shaped Output Shaft Size (Dia*Length)	6 x 14 mm
Gearbox Size (Dia*Length*Thick)	46 x 32 x 25 mm
Motor Size (Length*Max.D)	30.8 x 24.4 mm
Weight	142 grams
Motor Driver Specifications	
Driver Model	L298N
Driver Chip	Double H Bridge L298N
Motor Supply Voltage (Maximum)	46V
Motor Supply Current (Maximum)	2A
Logic Voltage	5V
Logical Current	0-36mA
Driver Voltage	5-35 V
Driver Current	2A

Maximum Power	25W
Motor Power Supply Specifications	
Rated Power	24W
Input Voltage	100-240V (AC) 50/60 HZ
Output	12V 2A (DC)
Circuit Power Supply Specifications	
Rated Power	5W
Input	100-240 V (AC) 50/60 HZ
Output	5V 1A (DC)

The electrical circuit diagram was made using Proteus Design Suite. In the Circuit, Arduino Uno controls and connects all the input devices including sensors, switches, potentiometers, and output devices including display, motor driver. Motor is powered via L298 driver because it can handle higher voltages and currents than Arduino microcontroller can allow. DC gear motor requires 12V 0.6A DC which cannot be provided through Arduino microcontroller pins that's why motor driver with a separate power supply is used to drive gear motor.

To control the motor speed, we used pulse width modulation (PWM) method. In PWM method voltage supply is, quickly and repeatedly, turned OFF for a fraction of cycle and then again turned ON for the rest of cycle. In this way average voltage delivered is reduced from the input voltage depending upon the amount of time the voltage was turned OFF, e.g., if voltage is turned ON for 50% of the time, then average voltage will drop to 6V from input voltage of 12V and motor speed will reduce to 50%.

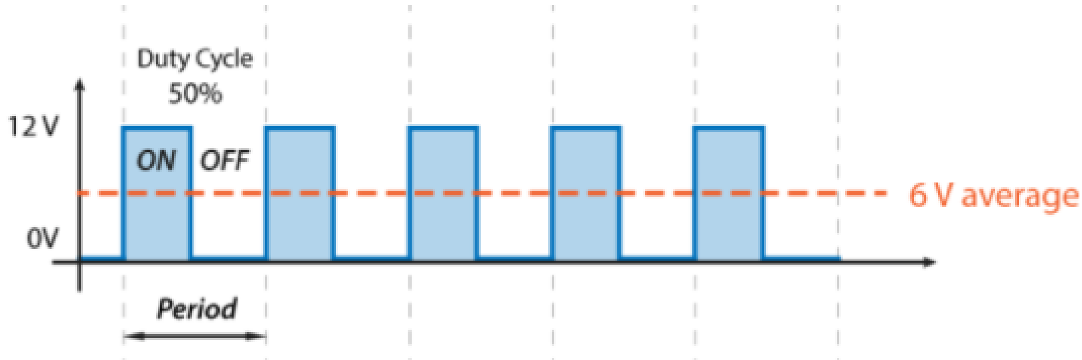


Figure 28: Working principle of pulse width modulation.

For variation of PWM frequency, we used a potentiometer that provides analog output to analog pin A0 according to the value set by user. According to the input from potentiometer, microcontroller will adjust the PWM frequency of its pin 7 that is connected to Enable A pin (input pin) of the motor driver, and thus changing the motor speed. Another latched acted button is attached to Arduino pin 0 which provides HIGH or LOW output according to its ON or OFF state set by user. Accordingly, Arduino flips the polarity of voltage provided to motor driver through pins 8 and 9, thus flips the motor's direction of rotation.

Digital hall effect sensors are used to measure speed of rotation shafts in mechanism. Two digital hall effect sensors provide HIGH input to Arduino pins 2 and 3, respectively, whenever a magnetic field of a certain strength is detected by them. Magnetic field strength for triggering the pulse can be calibrated by a screw in the sensor. These sensors can be easily mounted near any of the shaft in assembly whose speed is to be measured. For providing the magnetic field trigger to sensors we designed discs, with small magnets attached, to be mounted to desired shafts. Liquid Crystal display of 20x4 is used to show important parameters of operation. LCD get its input from Arduino via pins 1, 4, 5, 6, 11, 12.

CHAPTER 4: RESULTS AND DISCUSSIONS

For testing of the designed STEM KIT, we made several assemblies for gear train and 4-bar mechanisms.

Simple gear train as seen in the figure 29, is made by arranging spur 5 gears with 16, 20, 24, 28, 32 teeth, respectively. This arrangement results in the reduction ratio of 2.0. In simple gear train, each shaft inhabits 1 gear, is supported by sliding blocks. Sliding blocks will be tightened in place where gears will mesh perfectly. This design provides freedom to mesh any two gears from the option by just adjusting the distance between shafts by losing and tightening the 3mm screws that restricts the sliding movement of block inside side carriage beam. Gear couplings are used to mount gear to shaft that are designed with 4 holes to attach to gear and 1 hole to attach it to shaft. Upon tightening the screws gear gets locked on the shaft. Power is provided by a gear motor connected to the left most shaft by a gear coupling, power gets transferred to the shafts on right and reaches the fifth shaft with half the speed. Similar simple gear assemblies can be made with any combination of available gears.

For measurement of rpm value, we placed hall effect sensor near shaft 1 and 5. Also visible in figure 29 are two discs attached to shaft 1 and 5. Each of these discs has a small magnet attached to them, the shaft, disc, and magnet rotates and provides pulses to hall effect sensor form rpm calculation. Separate from the main base is the control panel that inhabits Arduino microprocessor, motor driver, lcd screen 20*4, potentiometer, and a switch. Sensors are attached to Arduino which calculated the rpm and displays on lcd.

Secondly, a compound gear assembly is made using 3 set of gears (figure 30). Each set has 20 and 30 teeth gears, resulting in 0.66 reduction ratio. 3 successive sets of 0.66 reduction provides over all 0.296 reduction. Each shaft has 2 gears mounded on it, one taking power and other transmitting it to next shaft. This demonstrates how compound gear arrangement can result in higher gear and reduction ratios. Two sensors are placed near shaft 1 and 4 to measure and compare their speeds.

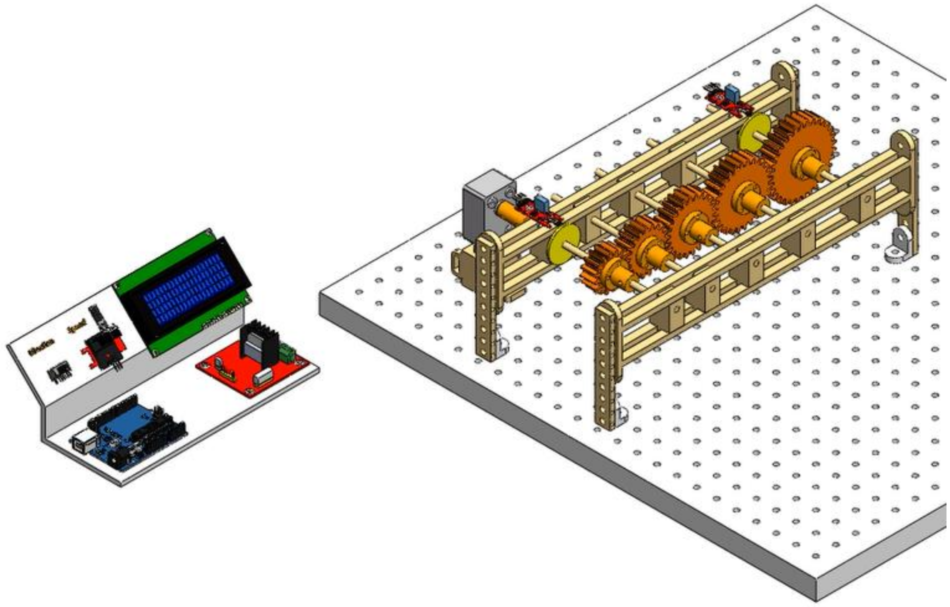


Figure 29: Simple gear train assembly with control panel.

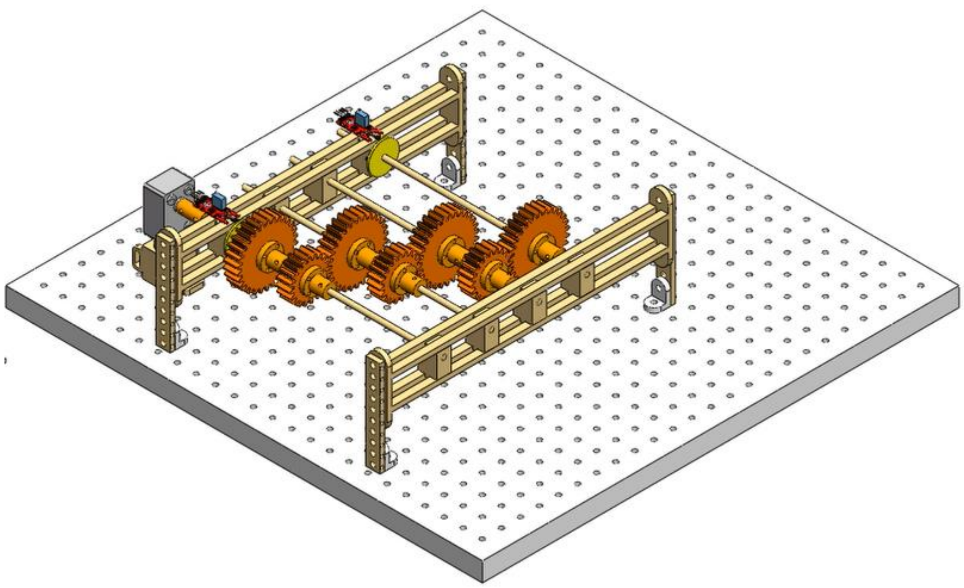


Figure 30: Compound gear train assembly.

Similarly, an assembly is made using bevel gears. In the assembly in figure 31 two sets of bevel gears are used to turn the power at 90° twice.

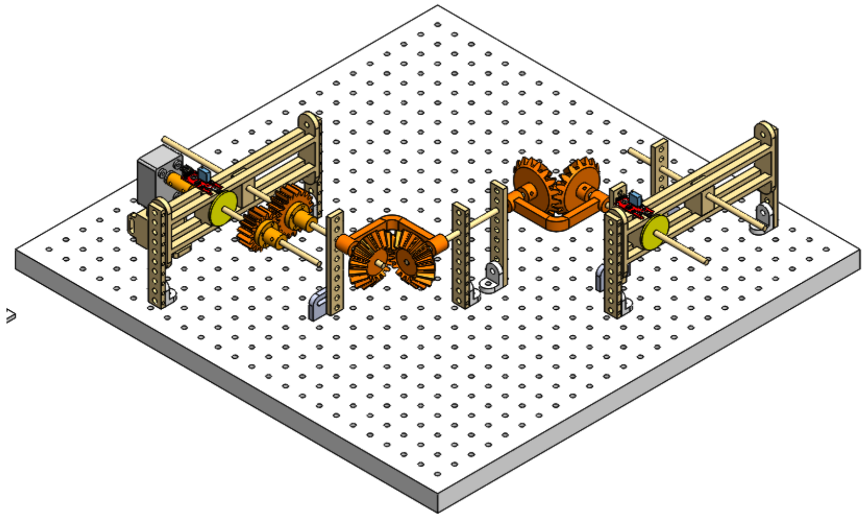


Figure 31: Bevel gear train assembly.

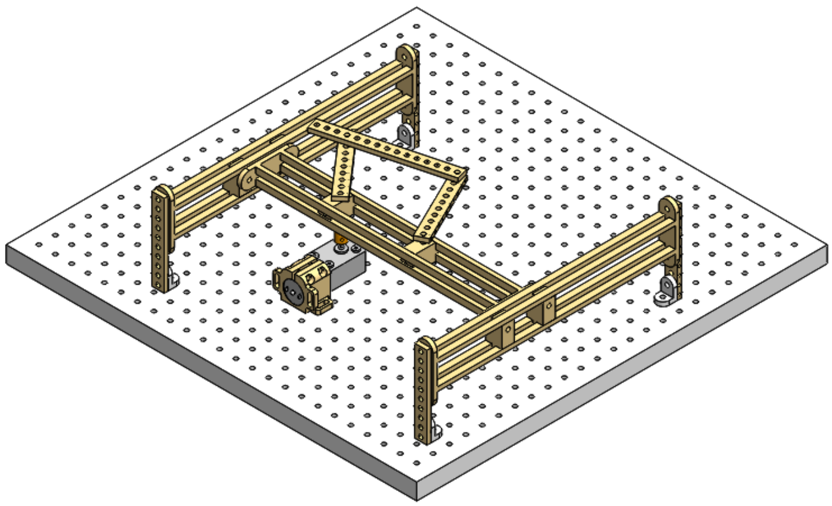


Figure 32: 4-bar crank-rocker assembly.

For testing of 4-bar mechanism, an assembly for crank-rocker configuration is made (figure 32). Two side carriage beams support elevate the mechanism above the motor. A side carriage beam, attached to the other two, is in inverted form with blocks fixed in place to act as ground link in 4-bar mechanism. Through one block passes the input shaft, on which crank link is attached. Though other block is attached the rocker link. Crank link and rocker link are attached to coupler link using circular billets that are screwed from side with one of the joining links.

Crank-slider mechanism is tested by making slight changes to the previous assembly. In this case, figure 33, second slider block is made free to move inside inverted side carriage beam to provide crank-slider behavior.

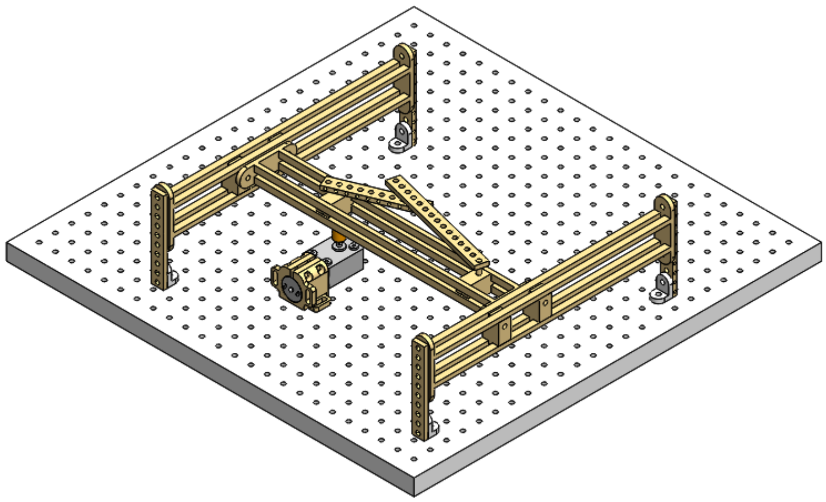


Figure 33: 4-bar crank-slider assembly.

The figures 34 below shows the electrical circuit simulated in Proteus Design Suite. All components are working as planned and LCD is showing that motor is in operation in anti-clockwise rotation, 55% of total input voltage in being provided, both sensors are providing inputs and Arduino is computing and showing RPMs. While simulating the circuit in Proteus, periodic magnetic field cannot be provided. To test the working of circuit, two input periodic signals are provided to Arduino pins 2 and 3. The figure 35 shows graph of signals of 1 Hz and 2 Hz that were provided as inputs to check our circuits and programs working. Our program in rightfully calculating RPM from these periodic signals who are mimicking the rotating disc with magnet attached.

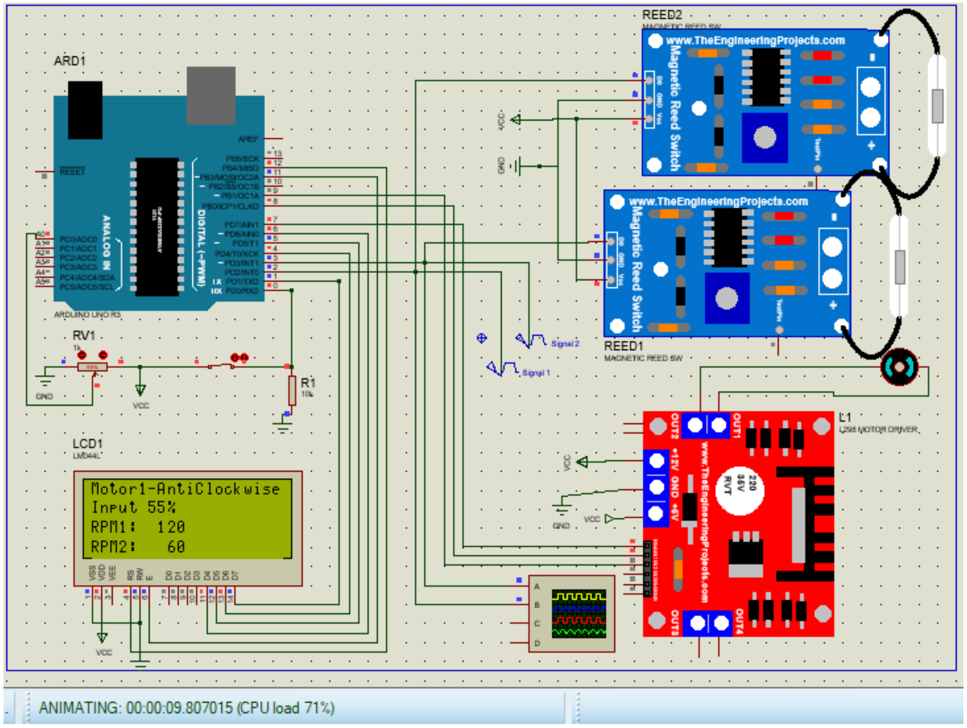


Figure 34: Electrical circuit simulation in Proteus

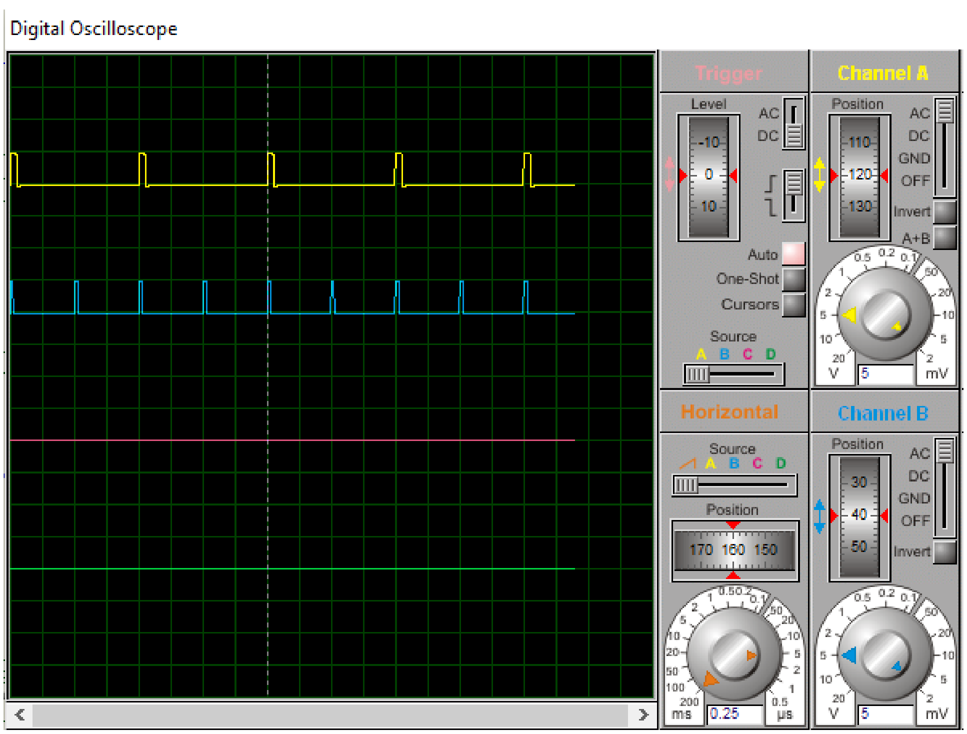


Figure 35: Input signals provided to circuit.

Table 5: Mechanical and Electrical components.

Component	Detail	Quantity Used
Base	500*500 mm	1x
Long Side carriage beam	300 mm	3x
Small Side carriage beam	180 mm	3x
Small Support beam	100 mm	12x
Large Support beam	150 mm	4x
Motor holder		1x
Spur Gear	16,18,20,22,24,26, 28,30,32,34,36 Teeth	2x each
Bevel gear	17 Teeth	4x
Shafts	260, 160, 50, 30 mm	6x, 6x, 4x, 4x
Shaft coupling		3x
Gear coupling		15x
Support brackets		12x
Geneva drive		1x
Magnetic disc		3x
Arduino	Uno R3	1x

Gear Motor	50 rpm	1x
Motor Driver	L298	1x
Magnetic field sensor		2x
LCD display	20*4 Alphanumeric	1x
Potentiometer	1k ohm	1x
Latched action Switch		1x
Resistor	1k ohm	1x
Power Supply	12V and 5V	2x

Table 5, summarizes all the mechanical and electrical parts of the STEM KIT necessary to implement the mechanisms discussed.

CHAPTER 5: CONCLUSION AND RECOMMENDATION

Concluding the project “Development of STEM KITS for engineering students”, we achieved basic and fundamental objectives successfully. We designed a platform which is portable, modular and multi-purpose, and offers digital measurements. Generally, modular and flexible platforms provide the option to be extended and expanded to include more features and capabilities. Similarly, our KIT performs different gear assemblies as well as different 4-bar configurations utilizing same parts. The Side carriage beam we designed is usable for placing and supporting gear shafts as well as making 4-bar mechanisms. We designed a motor support that can hold the motor in desired motion in horizontal as well as vertical direction. Support beams used in gear assemblies are also used as moving and rotation arms in 4-bar configurations. By fixing or allowing sliding of blocks inside side carriage beam we can get more than one function.

A basic electrical circuit is designed to get measurements from sensors, process them and display in the screen. Sensors and magnetic disc arrangement is such that it can be placed anywhere in the assembly to get speed measurements. Direction changing and speed varying capabilities are also provided to motor by the circuit. These features provides the targeted flexibility to users to some degree. Two motors can be attached with motor driver used. Appendix III contains the Arduino code used to operate electrical circuit.

There can be many improvements to the design to enable more experiments, also more components can be added to enhance experimental capability. Helical gear, worm gear, planetary gears can also be included to increase the diversity of gear train experiment. Many more mechanics related components can be added like; cam-follower, belt and pully, etc., to provide more options to the user. In 4-bar mechanism, further components or modifications can be made to enable more configurations like, double crank, double rocker, etc.,

In terms of electrical circuit, more powerful microprocessor can be used that can control more number of sensors for measurements like, displacement, speed, and acceleration etc. Arduino microprocessor we used has limited capability that it allows a limited number of attached devices and works in rigid way on the program provided to it. We implemented the concept using minimum parts to avoid complexity, and to proof the working. Electrical controller can be easily extended by experts of electrical and programming, to inhabit more user-friendly software, where users can give wide range of commands possible with modern operating systems and smart devices. As electronics is not a strong subject for mechanical engineering students, this section of the project can be greatly improved to include modern functions like, large screens, ability to export data, ability to connect new sensors, ability to calibrate them, etc.

REFERENCES

[1] Amazon Inc. 2021. *DC12V 2A Power Supply Adapter, SANSUN AC100-240V to DC12V Transformers, Switching Power Supply for 12V LED Strip Lights, 12 Volt 2 Amp Power Adaptor, 2.1mm X 5.5mm US Plug (1pcs)*. [online] Available at: < https://www.amazon.com/Adapter-SANSUN-AC100-240V-Transformers-Switching/dp/B01AZLA9XQ/1 > [Accessed 11 May 2021].

[2] Amazon Inc. n.d. *Greartisan DC 12V 50RPM Turbo Worm Geared Motor High Torque Turbine Worm Gear Box Reduction Motor 6mm Shaft JSX180-370*. [online] Available at: < https://www.amazon.com/dp/B086SM66SD/ref=redir_mobile_desktop?_encoding=UTF8&adId=200047393568711&id=6610047052441483&psc=1&qualifier=1622454862&ref_=syn_sd_onsite_mobileweb_186&spPl=1&th=1&uh_it=5cfe340468ad863c1f06ab575243db60 > [Accessed 10 May 2021].

[3] Arduino.cc. 2019. *Arduino - LiquidCrystal*. [online] Available at: < https://www.arduino.cc/en/Reference/LiquidCrystal> [Accessed 12 May 2021].

[4] Arduino.cc. 2021. *attachInterrupt() - Arduino Reference*. [online] Available at: < https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/ > [Accessed 11 May 2021].

[5] Arduino.cc. n.d. *micros() - Arduino Reference*. [online] Available at: < https://www.arduino.cc/reference/en/language/functions/time/micros/ > [Accessed 20 June 2021].

[6] Cadgill.blogspot.com. 2019. *Involute Teeth Profile Spur Gear Design*. [online] Available at: < https://cadgill.blogspot.com/2019/04/equation-driven-spur-gear-design.html > [Accessed 20 December 2020].

[7] Carroll, K. and Scott, C., 2017. Creating STEM Kits for the Classroom. *Science and Children*, 055(01).

[8] Daycounter.com. 2020. *Bevel Gear Calculator*. [online] Available at: < https://www.daycounter.com/Calculators/Bevel-Gear-Calculator.phtml > [Accessed 19 January 2021].

[9] Engineersedge.com., *Gear Design Engineering Tools Data Knowledge Menu - Engineers Edge*. [online] Available at: < https://www.engineersedge.com/gear_menu.shtml > [Accessed 2 February 2021].

[10] Engino Australia., *STEM MECHANICS: Gears & Worm drives | BY ENGINO*. [online] Available at: < https://www.engino.com.au/stem-mechanics-gears-worm-drives-by-engino > [Accessed 9 November 2020].

[11] Gray, W. and Albert, W., 2017. Create a STEM Pipeline for Students Who Become Engineering Majors Who Become Engineers

[12] Instructables.com. 2014. *Make Geneva Wheels of Any Size*. [online] Available at: < https://www.instructables.com/Make-Geneva-Wheels-of-Any-Size-in-a-Easier-Way/ > [Accessed 11 January 2021].

[13] Kanri, K., *Calculation of Gear Dimensions | KHK Gears*. [online] KHK Gears. Available at: < https://khkgears.net/new/gear_knowledge/gear_technical_reference/calculation_gear_dimensions.html > [Accessed 3 February 2021].

[14] Lewis, J., 2017. *7 Arduino LCD display tips and tricks - Bald Engineer*. [online] Bald Engineer. Available at: < https://www.baldengineer.com/arduino-lcd-display-tips.html > [Accessed 13 May 2021].

[15] Nasir, S., 2021. *Arduino Library for Proteus - The Engineering Projects*. [online] The Engineering Projects. Available at: < http://www.theengineeringprojects.com/2015/12/arduino-library-proteus-simulation.html > [Accessed 7 May 2021].

[16] Nasir, S., 2021. *Magnetic Reed Switch Library for Proteus - The Engineering Projects*. [online] The Engineering Projects. Available at: < https://www.theengineeringprojects.com/2018/05/magnetic-reed-switch-library-for-proteus.html > [Accessed 5 May 2021].

[17] Nasir, S., 2021. *New Proteus Libraries for Engineering Students - The Engineering Projects*. [online] The Engineering Projects. Available at: < https://www.theengineeringprojects.com/2016/08/new-proteus-libraries-engineering-students.html > [Accessed 8 May 2021].

[18] Regnerus, J., 2011. *How to create Spur and Helical Gears in SOLIDWORKS*. [online] SOLIDWORKS Tech Blog. Available at: < https://blogs.solidworks.com/tech/2014/07/how-to-create-spur-and-helical-gears-in-solidworks.html > [Accessed 28 January 2021].

[19] SDP/SI Small Mechanical Components. 2016. *Helical Gear Calculations, Crossed Helical Gear Meshes, and Bevel Gearing - SDPSI*. [online] Available at: < https://www.sdp-si.com/resources/elements-of-metric-gear-technology/page4.php#Section8 > [Accessed 17 January 2021].

[20] Stem-supplies.com. 2019., *Mechanical Engineering: Robotic Arms Kit - STEM Supplies*. [online] Available at: < https://www.stem-supplies.com/engineering/simple-and-complex-machines/mechanical-engineering-robotic-arms-kit > [Accessed 8 November 2020].

[21] TecQuipment., *Bar Linkages Kit ES15 | Engineering Science | TecQuipment*. [online] Available at: < https://www.tecquipment.com/bar-linkages-kit > [Accessed 8 November 2020].

APPENDIX I: SOLIDWORKS EQUATIONS FOR SPUR GEAR

Name	Value / Equation	Evaluates to	Comments
Global Variables			
"m"	= 2	2	
"N"	= 16	16	
"circular pitch"	= "m" * pi	6.28319	
"pitch dia"	= "N" * "m"	32	
"Outside dia"	= ("N" + "m") * "m"	36	
"Pressure angle"	= 20	20	
Add global variable			
Features			
Add feature suppression			
Equations			
"D2@Dias"	= "pitch dia"	32mm	
"D1@Dias"	= "Outside dia"	36mm	
"D1@Teeth"	= 90 / "N"	5.63deg	Halh-tooth angle
"D2@Teeth"	= "Pressure angle"	20deg	
"D1@Fillet1"	= "m" * 0.3	0.6mm	Fillet radius
"D1@CirPattern1"	= "N"	16	
"D2@Sketch4"	= 0.04 * "m"	0.08mm	Bachlash
"D1@Sketch4"	= 0.04 * "m"	0.08mm	Backlash
"D3@Dias"	= ("N" - (1.25 * "m")) * "m"	27mm	Root Dia

APPENDIX II: SOLIDWORKS EQUATIONS FOR BEVEL GEAR

Name	Value / Equation	Evaluates to	Comments
⊟ Global Variables			
"sigma"	= 90	90	shaft angle
"a"	= 20	20	pressure angle
"L"	= 15	15	face width
"m"	= 3mm	3mm	module
"z"	= 17	17	number of teeth
"dp"	= "m" * "z"	51	pitch diameter
"delta"	= 45	45	pitch angle
"tha"	= atn (2 * sin ("delta") / "z")	4.75543	addendum angle
"thf"	= atn (1.25 * 2 * sin ("delta") / "z")	5.93664	deddendum angle
"dpv"	= "dp" / cos ("delta")	72.1249	virtual pitch dia
"dbv"	= ("dp" * cos ("a")) / cos ("delta")	67.7752	virtual base dia
"dfv"	= ("dp" - 2.5 * "m" * cos ("delta")) / cos ("delta"	64.6249	virtual root dia
"dav"	= ("dp" + 2 * "m" * cos ("delta")) / cos ("delta")	78.1249	virtual addendum dia
"dfvu"	= IIF ("dbv" > "dfv" , "dfv" , "dbv" * 0.99)	64.6249mm	dfu undercut
"psi"	= 360 / (4 * "z" / cos ("delta"))	3.74351	half tooth angle
"inva"	= tan ("a") * 180 / pi - "a"	0.853958	involute of alpha
"t"	= sqr ("dav" ^ 2 / "dbv" ^ 2 - 1) * 1.01	0.579084mm	max involute param
Add global variable			
⊟ Features			
Add feature suppression			
⊟ Equations			
"D2@Sketch1"	= "tha"	4.76deg	
"D3@Sketch1"	= "thf"	5.94deg	
"D1@Sketch1"	= "delta"	45deg	
"D4@Sketch1"	= "dp" / 2	25.5mm	
"D6@Sketch1"	= "L"	15mm	
"D1@CirPattern1"	= "z"	17	
"D5@Sketch1"	= "dp" / 20	2.55mm	
"D1@Sketch2"	= "dbv"	67.78mm	
"D2@Sketch2"	= "dfvu"	64.62mm	
"D3@Sketch2"	= "psi"	3.74deg	

APPENDIX III: ARDUINO CODE

```
#include <Arduino.h>
#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 6, d5 = 5, d6 = 4, d7 = 1;
LiquidCrystal lcd (rs, en, d4, d5, d6, d7);

int enA = 7;
int in1 = 8;
int in2 = 9;
int button = 0;

const byte PulsesPerRevolution = 1;
const unsigned long ZeroTimeout = 10000000;

const byte numReadings = 2;                 // Number of samples for smoothing. The higher, the more
smoothing, but it's going to
                                            // react slower to changes. 1 = no smoothing. Default: 2.

volatile unsigned long LastTimeWeMeasured;  // Stores the last time we measured a pulse so we can calculate the
period.
volatile unsigned long PeriodBetweenPulses = ZeroTimeout+1000;
                                            // Stores the period between pulses in microseconds.

volatile unsigned long PeriodAverage = ZeroTimeout+1000;
                                            // Stores the period between pulses in microseconds in total

unsigned long FrequencyRaw;                 // Calculated frequency, based on the period.
unsigned long FrequencyReal;                // Frequency without decimals.
unsigned long RPM;                          // Raw RPM without any processing.
unsigned int PulseCounter = 1;              // Counts the amount of pulse readings we took so we can average
multiple pulses before calculating the period.

unsigned long PeriodSum;                              // Stores the summation of all the periods to do the
average.
unsigned long LastTimeCycleMeasure = LastTimeWeMeasured;  // Stores the last time we measure a pulse in that
cycle.
                              // We need a variable with a value that is not going to be affected by the interrupt
                                        // because we are going to do math and functions that are going to mess up if
the values
```

```
                                        // changes in the middle of the cycle.
unsigned long CurrentMicros = micros();         // Stores the micros in that cycle.


unsigned int AmountOfReadings = 1;
unsigned int ZeroDebouncingExtra;


// Variables for smoothing tachometer:
unsigned long readings[numReadings];          // The input.
unsigned long readIndex;                       // The index of the current reading.
unsigned long total;                           // The running total.
unsigned long average;                         // The RPM value after applying the smoothing.


//SECOND SENSOR
volatile unsigned long LastTimeWeMeasured2;   // Stores the last time we measured a pulse so we can calculate the
period.
volatile unsigned long PeriodBetweenPulses2 = ZeroTimeout+1000;  // Stores the period between pulses in
microseconds.
                               // It has a big number so it doesn't start with 0 which would be interpreted as a high
frequency.
volatile unsigned long PeriodAverage2 = ZeroTimeout+1000;
                              // Stores the period between pulses in microseconds in total, if we are taking multiple
                    pulses.
                              // It has a big number so it doesn't start with 0 which would be interpreted as a high
frequency.
unsigned long FrequencyRaw2;          // Calculated frequency, based on the period. This has a lot of extra decimals
without the decimal point.
unsigned long FrequencyReal2;          // Frequency without decimals.
unsigned long RPM2;                    // Raw RPM without any processing.
unsigned int PulseCounter2 = 1;        // Counts the amount of pulse readings we took so we can average multiple
                                       pulses before calculating the period.


unsigned long PeriodSum2;              // Stores the summation of all the periods to do the average.
unsigned long LastTimeCycleMeasure2 = LastTimeWeMeasured2;  // Stores the last time we measure a pulse in that
cycle.
                                        // We need a variable with a value that is not going to be affected by the
interrupt
                                        // because we are going to do math and functions that are going to mess up if
                                        the values
                                        // changes in the middle of the cycle.
unsigned long CurrentMicros2 = micros();  // Stores the micros in that cycle.
                                            // We need a variable with a value that is not going to be affected by the
interrupt
```

```
                                                    // because we are going to do math and functions that are going to mess up
                                                    if the values
                                                            // changes in the middle of the cycle.
unsigned int AmountOfReadings2 = 1;
unsigned int ZeroDebouncingExtra2;  // Stores the extra value added to the ZeroTimeout to debounce it.
                                                  // The ZeroTimeout needs debouncing so when the value is close to the
threshold it
                                            // doesn't jump from 0 to the value. This extra value changes the threshold a little
                                                  // when we show a 0.
                                          // Variables for smoothing tachometer:
unsigned long readings2[numReadings];           // The input.
unsigned long readIndex2;                       // The index of the current reading.
unsigned long total2;                           // The running total.
unsigned long average2;                         // The RPM value after applying the smoothing


void setup() {
  lcd.begin(20, 4);
  lcd.print(" ExperiPad STEM KIT ");


  attachInterrupt(digitalPinToInterrupt(2), Pulse_Event, RISING);  // Enable interruption pin 2 when going from LOW
to HIGH.
  delay(1000);
  attachInterrupt(digitalPinToInterrupt(3), Pulse_Event2, RISING);  // Enable interruption pin 2 when going from LOW
to HIGH.
  delay(1000);
}


void loop() {


  int potValue = analogRead(A0);
  int pwmOutput = map(potValue, 0, 1023, 0, 255);
  analogWrite(enA, pwmOutput);
  lcd.setCursor(0, 1);
  lcd.print("Input ");
  lcd.print(map(potValue, 0, 1023, 0, 100));
  lcd.print("%");


  if(digitalRead(button) == LOW){
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
     lcd.setCursor(0, 0);
```

```
    lcd.print("Motor1-Clockwise    ");
  }
  if(digitalRead(button) == HIGH){
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
     lcd.setCursor(0, 0);
     lcd.print("Motor1-AntiClockwise");
  }

  LastTimeCycleMeasure = LastTimeWeMeasured;                              // Store the LastTimeWeMeasured in a variable.
  CurrentMicros = micros();                                               // Store the micros() in a variable.

                          // CurrentMicros should always be higher than LastTimeWeMeasured, but in rare occasions that's
not true.
                            // I'm not sure why this happens, but my solution is to compare both and if CurrentMicros is lower
than
                          // LastTimeCycleMeasure I set it as the CurrentMicros.
                           // The need of fixing this is that we later use this information to see if pulses stopped.
  if(CurrentMicros < LastTimeCycleMeasure)
  {
    LastTimeCycleMeasure = CurrentMicros;
  }

  // Calculate the frequency:
  FrequencyRaw = 10000000000 / PeriodAverage;               // Calculate the frequency using the period between pulses.

                                                       // Detect if pulses stopped or frequency is too low, so we can show 0
Frequency:
  if(PeriodBetweenPulses > ZeroTimeout - ZeroDebouncingExtra || CurrentMicros - LastTimeCycleMeasure >
ZeroTimeout - ZeroDebouncingExtra)
  {                                                    // If the pulses are too far apart that we reached the timeout for zero:
    FrequencyRaw = 0;  // Set frequency as 0.
    ZeroDebouncingExtra = 2000;           // Change the threshold a little so it doesn't bounce.
  }
  else
  {
    ZeroDebouncingExtra = 0;                              // Reset the threshold to the normal value so it doesn't bounce.
  }

  FrequencyReal = FrequencyRaw / 10000;  // Get frequency without decimals.
                                                        // This is not used to calculate RPM but we remove the decimals just in case
```

```
                                              // you want to print it.
                                             // Calculate the RPM:
  RPM = FrequencyRaw / PulsesPerRevolution * 60;  // Frequency divided by amount of pulses per revolution multiply
by
                                                                  // 60 seconds to get minutes.
  RPM = RPM / 10000;                          // Remove the decimals.

  // Smoothing RPM:
  total = total - readings[readIndex];          // Advance to the next position in the array.
  readings[readIndex] = RPM;                            // Takes the value that we are going to smooth.
  total = total + readings[readIndex];          // Add the reading to the total.
  readIndex = readIndex + 1;                    // Advance to the next position in the array.

  if (readIndex >= numReadings)                // If we're at the end of the array:
  {
    readIndex = 0;                              // Reset array index.
  }

  // Calculate the average:
  average = total / numReadings;               // The average value it's the smoothed result.

  // Print information on the serial monitor:
  // Comment this section if you have a display and you don't need to monitor the values on the serial monitor.
  // This is because disabling this section would make the loop run faster.

  lcd.setCursor(0, 2);
  char buff[20];
  sprintf(buff, "RPM1: %4i   ", average);
  lcd.print(buff);

  LastTimeCycleMeasure2 = LastTimeWeMeasured2;            // Store the LastTimeWeMeasured in a variable.
  CurrentMicros2 = micros();                               // Store the micros() in a variable.

  if(CurrentMicros2 < LastTimeCycleMeasure2)
  {
    LastTimeCycleMeasure2 = CurrentMicros2;
  }

 // Calculate the frequency:
```

```
  FrequencyRaw2 = 10000000000 / PeriodAverage2;  // Calculate the frequency using the period between pulses.

                                                                                    // Detect if pulses stopped or frequency is too
                                                                                    low, so we can show 0 Frequency:
  if(PeriodBetweenPulses2 > ZeroTimeout - ZeroDebouncingExtra2 || CurrentMicros2 - LastTimeCycleMeasure2 >
ZeroTimeout - ZeroDebouncingExtra2)
  {                                             // If the pulses are too far apart that we reached the timeout for zero:
    FrequencyRaw2 = 0;  // Set frequency as 0.
    ZeroDebouncingExtra2 = 2000;                // Change the threshold a little so it doesn't bounce.
  }
  else
  {
    ZeroDebouncingExtra2 = 0;                   // Reset the threshold to the normal value so it doesn't bounce.
  }

  FrequencyReal2 = FrequencyRaw2 / 10000;       // Get frequency without decimals.
                                                // This is not used to calculate RPM but we remove the decimals just
in case
                                                // you want to print it.
                                                // Calculate the RPM:
  RPM2 = FrequencyRaw2 / PulsesPerRevolution * 60;  // Frequency divided by amount of pulses per revolution
multiply by
                                                                  // 60 seconds to get minutes.
  RPM2 = RPM2 / 10000;                                   // Remove the decimals.

  // Smoothing RPM:
  total2 = total2 - readings2[readIndex2];  // Advance to the next position in the array.
  readings2[readIndex2] = RPM2;  // Takes the value that we are going to smooth.
  total2 = total2 + readings2[readIndex2];  // Add the reading to the total.
  readIndex2 = readIndex2 + 1;  // Advance to the next position in the array.

  if (readIndex2 >= numReadings)  // If we're at the end of the array:
  {
    readIndex2 = 0;  // Reset array index.
  }

  // Calculate the average:
  average2 = total2 / numReadings;  // The average value it's the smoothed result.

  lcd.setCursor(0, 3);
  char buff2[20];
```

```
  sprintf(buff2, "RPM2: %4i   ", average2);
  lcd.print(buff2);

}

void Pulse_Event() { // The interrupt runs this to calculate the period between pulses:

  PeriodBetweenPulses = micros() - LastTimeWeMeasured;  // Current "micros" minus the old "micros" when the last pulse happens.
                                                        // This will result with the period (microseconds) between both pulses.
                                                        // The way is made, the overflow of the "micros" is not going to cause any issue.
  LastTimeWeMeasured = micros();      // Stores the current micros so the next time we have a pulse we would have something to compare with.

  if(PulseCounter >= AmountOfReadings)              // If counter for amount of readings reach the set limit:
  {
    PeriodAverage = PeriodSum / AmountOfReadings;         // Calculate the final period dividing the sum of all readings by the
                                                          // amount of readings to get the average.
    PulseCounter = 1;                   // Reset the counter to start over. The reset value is 1 because its the minimum setting allowed (1 reading).
    PeriodSum = PeriodBetweenPulses;  // Reset PeriodSum to start a new averaging operation.



    // Change the amount of readings depending on the period between pulses.
    // To be very responsive, ideally we should read every pulse. The problem is that at higher speeds the period gets
    // too low decreasing the accuracy. To get more accurate readings at higher speeds we should get multiple pulses and
    // average the period, but if we do that at lower speeds then we would have readings too far apart (laggy or sluggish).
// To have both advantages at different speeds, we will change the amount of readings depending on the period between pulses.
    // Remap period to the amount of readings:
    int RemapedAmountOfReadings = map(PeriodBetweenPulses, 40000, 5000, 1, 10);  // Remap the period range to the reading range.
    // 1st value is what are we going to remap. In this case is the PeriodBetweenPulses.
    // 2nd value is the period value when we are going to have only 1 reading. The higher it is, the lower RPM has to be to reach 1 reading.
    // 3rd value is the period value when we are going to have 10 readings. The higher it is, the lower RPM has to be to reach 10 readings.
    // 4th and 5th values are the amount of readings range.
    RemapedAmountOfReadings = constrain(RemapedAmountOfReadings, 1, 10);  // Constrain the value so it doesn't go below or above the limits.
```

```
    AmountOfReadings = RemapedAmountOfReadings;  // Set amount of readings as the remaped value.
  }
  else
  {
    PulseCounter++;  // Increase the counter for amount of readings by 1.
    PeriodSum = PeriodSum + PeriodBetweenPulses;  // Add the periods so later we can average.
  }

}  // End of First Pulse_Event.

void Pulse_Event2() { // The interrupt runs this to calculate the period between pulses:

  PeriodBetweenPulses2 = micros() - LastTimeWeMeasured2;  // Current "micros" minus the old "micros" when the last pulse happens.
                                              // This will result with the period (microseconds) between both pulses.
                                              // The way is made, the overflow of the "micros" is not going to cause any issue.
  LastTimeWeMeasured2 = micros();  // Stores the current micros so the next time we have a pulse we would have something to compare with.

  if(PulseCounter2 >= AmountOfReadings2)  // If counter for amount of readings reach the set limit:
  {
    PeriodAverage2 = PeriodSum2 / AmountOfReadings2;  // Calculate the final period dividing the sum of all readings by the
                                            // amount of readings to get the average.
    PulseCounter2 = 1;  // Reset the counter to start over. The reset value is 1 because its the minimum setting allowed (1 reading).
    PeriodSum2 = PeriodBetweenPulses2;  // Reset PeriodSum to start a new averaging operation.


    // Change the amount of readings depending on the period between pulses.
    // To be very responsive, ideally we should read every pulse. The problem is that at higher speeds the period gets
    // too low decreasing the accuracy. To get more accurate readings at higher speeds we should get multiple pulses and
    // average the period, but if we do that at lower speeds then we would have readings too far apart (laggy or sluggish).
    // To have both advantages at different speeds, we will change the amount of readings depending on the period between pulses.
    // Remap period to the amount of readings:
    int RemapedAmountOfReadings2 = map(PeriodBetweenPulses2, 40000, 5000, 1, 10);  // Remap the period range to the reading range.
    // 1st value is what are we going to remap. In this case is the PeriodBetweenPulses.
    // 2nd value is the period value when we are going to have only 1 reading. The higher it is, the lower RPM has to be to reach 1 reading.
```

```
    // 3rd value is the period value when we are going to have 10 readings. The higher it is, the lower RPM has to be to reach 10 readings.

    // 4th and 5th values are the amount of readings range.

    RemapedAmountOfReadings2 = constrain(RemapedAmountOfReadings2, 1, 10);  // Constrain the value so it doesn't go below or above the limits.

    AmountOfReadings2 = RemapedAmountOfReadings2;  // Set amount of readings as the remaped value.

  }

  else

  {

    PulseCounter2++;  // Increase the counter for amount of readings by 1.

    PeriodSum2 = PeriodSum2 + PeriodBetweenPulses2;  // Add the periods so later we can average.

  }


}  // End of Second Pulse_Event.
```