



BUILDING A DIGITAL TWIN OF SCREW JACK FOR PREDICTIVE MAINTANENCE

A Project report submitted

in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

in

MECHANICAL ENGINEERING

by

Y. Chandra Mouli
(19331A03G9)

Y. Srinivasulu
(20335A0321)

Y. Ganesh
(19331A03G5)

P. Vikas Mouli
(19331A03C4)

Under the esteemed guidance of

Dr Praveen Kalla

Associate Professor



DEPARTMENT OF MECHANICAL ENGINEERING

MAHARAJ VIJAYARAM GAJAPATHI RAJ COLLEGE ENGINEERING

(Autonomous)

(Approved by AICTE0, New Delhi, accredited by NBA of AICTE, NAAC with 'A' grade of UGC and permanently affiliated to Jawaharlal Nehru Technological University, Kakinada, Listed u/s 2(f) & 12(B) of the UGC act 1956)

VIZIANAGARAM

2019-2023



DEPARTMENT OF MECHANICAL ENGINEERING
MAHARAJ VIJAYARAM GAJAPATHI RAJ COLLEGE OF ENGINEERING
(Autonomous)



CERTIFICATE

This is to certify that the thesis entitled **“BUILDING A DIGITAL TWIN OF SCREW JACK FOR PREDICTIVE MAINTANENCE”** that is being submitted by Y. Chandra Mouli (19331A03G9), Y. Srinivasulu (20335A0321), Y. Ganesh (19031A03G5), P. Vikas Mouli (19031A03C4) in partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology in Mechanical Engineering** is a record of bonafide work done by her/him under my/our supervision during the academic year 2019 - 2023.

Dr. Praveen Kalla

Associate Professor
Dept. of Mechanical Engg.,
MVGR College of Engineering,
Vizianagaram.

Dr. S. Adinarayana

Professor & HoD
Dept. of Mechanical Engg.,
MVGR College of Engineering,
Vizianagaram.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We, the members of the project, heartfully thank our guides **Dr Praveen Kalla, Associate professor, MVGRCE** for his suggestions and co-operation in the completion of the project work precisely. We shall always cherish our association with his encouraging approachability, explanation, and freedom of thought and action.

We are deeply indebted to **Dr S. Adi Narayana**, Head of the department of mechanical engineering, for her constant encouragement for the completion of our project successfully. We thank the PRC Members for their suggestions and corrections. We thank all the Teaching and Non-Teaching staff of the department of mechanical engineering

It is a great privilege for us to convey our sincere gratitude to **Dr K V L Raju**, the principal for providing us with excellent lab facilities.

We are thankful to our family members who co-operated with us in this journey. We extend our special thanks to Shri M Chaitanya, MD & CTO Bot Forge Robotics Pvt Ltd, who was so easily approachable, helpful, and a source of encouragement during this span of work.

Project Associates

Y. Chandra Mouli -19331A03G9

Y. Srinivasulu -20335A0321

Y. Ganesh -19331A03G5

P. Vikas Mouli -19331A03C4

ABSTRACT

The concept of digital twin technology is one that is just starting to gain traction in business and, more recently, academics. Industry 4.0 has grown more quickly thanks to conceptual advances, especially in the manufacturing sector. There are many definitions of the Digital Twin, but the best described as the seamless integration of data going either way between a physical and virtual machine. We describe the issues, uses, and supporting technologies for artificial intelligence, the internet of things (IoT), and digital twins. A categorical review of recent studies is produced after a review of publications on digital twins. Manufacturing, healthcare, and smart cities have been categorised by study fields in the review, which also discusses a variety of publications that reflect these domains and the current level of research. The ability to unleash various potentials of virtual product creation has increased with the use and sophistication of simulation models. While simulation software has gotten increasingly powerful, attempts to maximize the potential benefits of product simulations are increasingly centered on the input data that is now accessible. More accurate simulation models that ultimately produce superior products can be created if it is able to incorporate use phase data from real-world applications into the simulation models or use this data to check simulation findings. This project enables the real time simulation and stress analysis of a screw jack in working condition.

Keywords:

Digital Twins, Applications, Enabling Technologies, Industrial Internet of Things (IIoT), Internet of Things (IoT), Machine Learning, Deep Learning, Literature Review

COURSE OUTCOMES

CO1 Students can develop their analytical skills and demonstrate their ability to use engineering analysis to inform design decisions.

CO2 Students can demonstrate their ability to apply design principles and techniques to real-world problems.

CO3 Students can develop their research skills, including literature review, data analysis, and report writing.

CO4 Students can develop their teamwork skills, including collaboration, communication, and leadership.

CO5 Students can learn about the importance of ethical conduct, safety, and sustainability in engineering practice

CO6 Students develop their communication skills, including technical writing and public speaking.

PROJECT WORK MAPPING WITH COs, POs AND PSOs:

CO	Program outcomes												PSOs	
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	2	2	2	3	3	1	1	2	3	1	3	3	2	2
CO2	2	1	2	2	3	1	1	2	2	1	2	3	1	3
CO3	1	2	2	3	2	2	2	2	3	2	3	2	2	3
CO4	2	3	3	2	3	2	2	3	3	2	2	3	3	2
CO5	3	2	2	3	2	1	1	3	2	2	2	3	2	2
CO6	2	2	3	2	3	2	2	2	2	1	2	2	1	2

Attainment Levels: High - 3; Medium - 2; Low – 1

PROGRAMME OUTCOMES:

PO	PROGRAM OUTCOMES
1	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to Complex engineering activities with an understanding of the limitations.
6	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7	Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11	Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12	Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOMES (ME) :

PSO	Statement
1	The student will be able to demonstrate the skill set required to suit the requirements of the industry as well a research environment
2	The student will be able to design experiments, conduct experiment, analyze, comprehend and report based on the knowledge acquired by the experience-based learning.

CONTENTS

TITLE PAGE	i
CERTIFICATE	ii
ACKNOWLEDGE	iii
ABSTRACT	iv
COURSE OUTCOMES	v
PROJECT WORK MAPPING WITH POs AND PSOs	vi
CONTENTS	vii
LIST OF FIGURES	vii

Chapter 1. INTRODUCTION

1.1. Introduction to Digital Twin Technology	2
1.2. Process of Digital Twin	3
1.3. Simulations vs. digital twins	3
1.4. Classification of Digital twin	4
1.5. The advantages and disadvantages of digital twins	5
1.6. Sensors	
1.6.1. Introduction to sensors	5
1.6.2. Working of a sensor	6
1.6.3. Classification of sensors based on material	7
1.6.4. Classification of sensors based on functionality	8
1.6.5. Advantages of Sensors	10
1.7. Introduction of Visual Studio Code	
1.7.1. Features of Visual Studio Code	11

1.8. Introduction to Arduino	12
1.8.1. Arduino UNO Board	13
1.8.2. Microcontroller	15
1.9. Introduction to Java Script	
1.9.1. Features of Java Script	17
1.10. Introduction to Three.js	
18	
1.11. Introduction to Fusion 360	
1.11.1. History of fusion 360	19
1.11.2. Features of Fusion 360	19
1.12. Screw Jack	
1.12.1. Types of screw jacks	21
1.12.1.1. Machine/worm gear screw jack	21
1.12.1.2. Ball screw jack	22
1.12.1.3. Bevel screw jack	22
1.12.2. Advantages of screw jack	
1.12.2.1. Mechanical Advantage	23
1.12.3. Limitations	23
1.13. Load Cell	24
1.14. Magnetic Encoder	25
1.15. Vue Framework	26
1.16. Electron-Java Script Library	
 Chapter 2. REVIEW OF LITERATURE	
2.1. Digital twin technology	28
2.2. Three.js	30
2.3. Screw jack	31
 Chapter 3. DESIGN AND DEVELOPMENT OF DIGITAL TWIN:	
3.1. Development of Digital twin of a screw jack using Processing	
application and an Arduino	32

3.2. Development of Digital twin of a screw jack using Java script, three.js, fusion 360 and webapp.	50
3.2.1. Experimental setup	
3.2.1.1. Construction.	51
3.2.1.2. Building a sensor box.	51
3.2.2. Designing and modelling	
3.2.2.1. URDF format of screw jack	52
3.2.3. Executing the web application	56
3.2.4. Procedure to execute	69
 Chapter 4. RESULTS AND DISCUSSION:	
4.1. Result of Digital Twin model using Arduino and Processing Application	71
4.2. Result of Digital Twin of a screw jack developed in fusion 360 and implemented in webapp.	73
4.2.1. Final result of Digital twin of screw jack	75
 Chapter 5. CONCLUSION	
5.1. Conclusion	77
REFERENCES	78

LIST OF FIGURES

Fig.No.	Description	Page No.
1.1	Reference of Digital Twin	2
1.2	Architecture of Digital Twin system	3
1.3	Simulation vs Digital Twin	4
1.4	Architecture of IoT	6
1.5	Model of Inductive sensor	7
1.6	Model of Capacitive sensor	7
1.7	Model of Photoelectric sensor	7
1.8	Model of Ultrasonic sensor	7
1.9	Visual Studio and Visual Studio code	11
1.10	Arduino IDE Interface	13
1.11	Arduino UNO Pinout Diagram	14
1.12	Java script Reference	16
1.13	Object in 3d space	18
1.14(a)	Fusion 360 (Model 1)	20
1.14(b)	Fusion 360 (Model 1)	20
1.15(a)	Machine/worm screw jack	21
1.15(b)	Ball screw jack	22
1.15(c)	Bevel screw jack	22
1.16	Load cell	24
1.17	Wheat stone bridge	24
1.18	Magnetic encoder	25

1.19	Architecture of Vue Frame work	25
3.1	MPU 6050 Sensor	32
3.2	Arduino UNO Board	32
3.3	Sensor and Arduino Board Integration	32
3.4(a)	Integration with the system (Ideal case)	33
3.4(b)	Integration with the system (Connected case)	33
3.5	Interface of Arduino IDE	34
3.6	Interface of Arduino IDE representing boards	34
3.7(a)	Compile button interface	43
3.7(b)	Pop up (Compiling done message)	44
3.8	Processing application logo	44
3.9	Interface of processing application	45
3.10	Execute button interface	49
3.11	Space containing 3d model	49
3.12	Screw jack with 3d printed	50
4.1	Digital twin of physical object in virtual space	71
4.2	Left orientation of 3d object	71
4.3	Right orientation of 3d object	72
4.4	Top view of 3d object	72
4.5	Bottom view of 3d object	72
4.6	Front view of screw jack developed in fusion 360	73
4.7	3d view of screw jack developed in fusion 360	73
4.8	Sensor box	74
4.9	Cross sectional view of sensor box consisting of M274 encoder, 5kg load cell hx711 module	74
4.10	Experimental setup of physical screw jack and its digital twin in ideal case	75

4.11	Experimental setup of physical screw jack and its digital twin when load is acting	75
------	--	----



CHAPTER 1

INTRODUCTION

1.1 Introduction to Digital Twin Technology

A virtual reproduction of a physical system or process, such as a machine, a structure, or an entire city, is known as a digital twin. It is a digital model that may be used to mimic and examine the functionality and behaviour of the system in the actual world. Digital twins have been around for a while, but recent developments in technology like the Internet of Things (IoT), artificial intelligence, and cloud computing have made it simpler to build and use them.

The physical system's physical and functional properties are captured by the digital twin, which is created to be an exact digital replica of the physical system. It is regularly updated with real-time data from sensors and other sources, which enables it to correctly depict the actual status of the physical system.

Users can learn more about the operation of the physical system and pinpoint areas for improvement by examining the data gathered from the digital twin.

Many different businesses, including manufacturing, healthcare, and transportation, can benefit from the use of digital twins. Digital twins, for instance, can be utilised to streamline production processes, raise product quality, and decrease downtime in the manufacturing industry. To model the human body and provide individualised treatment strategies in healthcare, digital twins can be used. Digital twins can be utilised in the transportation sector to track, improve, and lessen traffic congestion.

The digital twin is an effective technique that can help businesses streamline their processes, cut costs, and increase efficiency.



Fig 1.1 Reference of Digital Twin

1.2 Process of Digital Twin

A virtual representation created to faithfully represent a physical object is called a digital twin. The object being researched, such as a wind turbine, is equipped with a variety of sensors that are connected to key functioning regions. These sensors generate information about a variety of performance characteristics of the physical device, including energy output, temperature, environmental conditions, and more. The processing system then applies this information to the digital copy.

The virtual model can then be used to run simulations, analyse performance problems, and suggest potential modifications in order to get useful insights that can then be applied to the real physical device

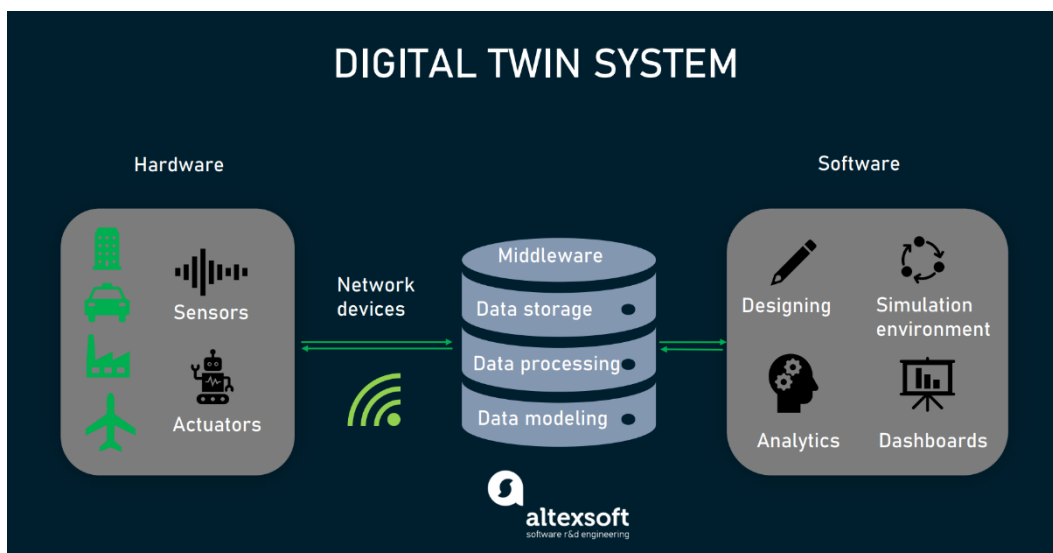


Fig.1.2. Architecture of Digital Twin system.

1.3 Simulations vs. digital twins

Although both simulations and digital twins use digital models to replicate a system's various processes, a digital twin is actually a virtual environment, making it far more rich for research. The primary distinction between digital twin and simulation is one of scale: While a simulation typically studies a single process, a digital twin can run an unlimited number of useful simulations to study multiple processes.

The distinctions do not stop there. For example, simulations rarely benefit from real-time data. However, digital twins are built around a two-way information flow that begins when object sensors provide relevant data to the system processor and continues when the processor's insights are shared back with the origin source user

Digital twins are able to study more issues from far more vantage points than standard simulations can because they have better and constantly updated data related to a wide range of areas, combined with the added computing power that comes with a virtual environment — with greater ultimate potential to improve products and processes

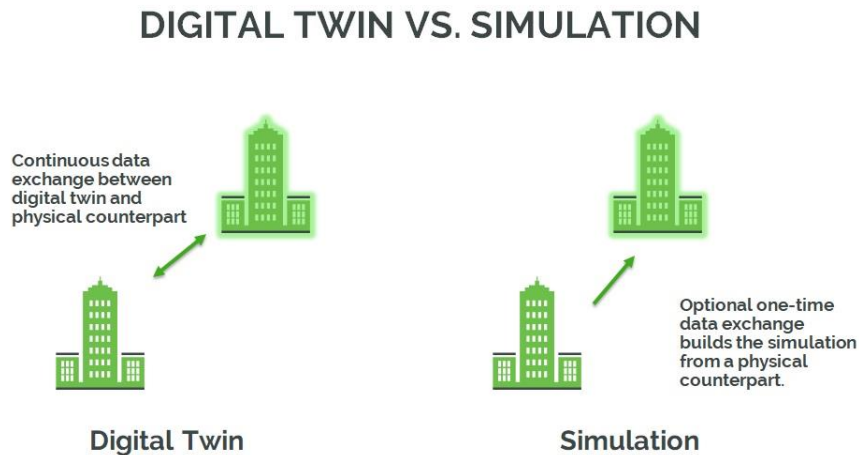


Fig 0.3. Simulation vs Digital Twin

1.4 Classification of Digital twin

Depending on the extent of product magnification, there are many sorts of digital twins. The primary distinction between these twins is their field of application. Several sorts of digital twins frequently coexist within a system or process. Let's look over the many sorts of digital twins to understand the differences and how they are used

- **Parts twins/Component twins**

Component twins are the fundamental unit of a digital twin, representing the smallest example of a working component. Components twins are roughly the same thing, except they refer to significantly less important components.

- **Twins of assets**

An asset is formed when two or more components work together to form a unit. Asset twins allow you to investigate the interaction of those components, resulting in a wealth of performance data that can be analysed and transformed into meaningful insights.



- **System twins or Unit twins**

The next degree of magnification involves system or unit twins, which allow you to see how various assets interact to build a fully functional system. System twins provide visibility into asset interactions and may identify performance improvements.

- **Twins in the process**

Process twins, the macro level of magnification, show how systems interact to produce an entire manufacturing complex. Are all of those systems synced to run at top efficiency, or will delays in one system have an impact on others? Process twins can assist in determining the specific timing schemes that influence overall effectiveness.

1.5 The advantages and disadvantages of digital twins

- **Improved R&D**

The utilisation of digital twins allows for more effective product research and creation, with an abundance of data generated concerning expected performance outcomes. This data can lead to insights that can help businesses make necessary product improvements before going into production.

- **Increased effectiveness**

Even after a new product has entered production, digital twins can assist in mirroring and monitoring production systems in order to achieve and maintain optimal efficiency throughout the manufacturing process.

- **Product obsolescence**

Digital twins can even assist producers in determining what to do with products that have reached the end of their product lifecycle and require final processing, such as recycling or other measures. They can identify which product materials can be used by employing digital twins

1.6 Sensors

1.6.1 Introduction to sensors

A sensor is a device that detects and responds to some type of input from the physical environment. The input can be light, heat, motion, moisture, pressure or any number of other environmental phenomena. The output is generally a signal that is converted to a human-readable display at the sensor location or transmitted electronically over a network for reading or further processing.

Sensors play a pivotal role in the internet of things (IoT). They make it possible to create an ecosystem for collecting and processing data about a specific environment so it can be monitored, managed and controlled more easily and efficiently. IoT sensors are used in homes, out in the field, in automobiles, on airplanes, in industrial settings and in other environments. Sensors bridge the gap between the physical world and logical world, acting as the eyes and ears for a computing infrastructure that analyses and acts upon the data collected from the sensors.

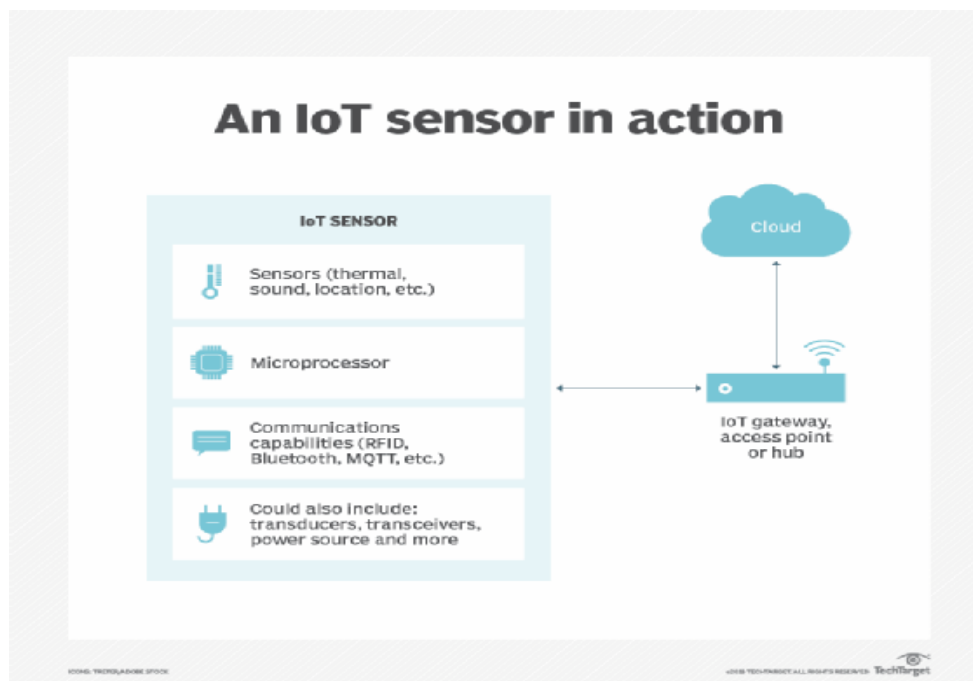


Fig 1.4 Architecture of IoT

1.6.2 Working of a sensor

A sensor converts the physical action to be measured into an electrical equivalent and processes it so that the electrical signals can be easily sent and further processed. The sensor can output whether an object is present or not present (binary) or what measurement value has been reached (analog or digital).

Components of a sensor

- The sensing section contains the sensor itself which is based on a particular technology. The variety of technologies means you can select a sensor technology which fits your application.
- The processing circuitry converts the physical variable into an electrical variable.
- The signal output contains the electronics connected to a control system.

1.6.3 Classification of sensors based on material

Types of Sensors:

- **Inductive sensors**

Inductive sensors generate an electromagnetic field. This in turn generates eddy currents in objects made of metal. The sensor detects this change.

- **Capacitive sensors**

Capacitive sensors generate a capacitive measuring field. An entering object results in a change to the measuring field. The sensor responds to this change.

- **Photoelectric sensors**

Photoelectric sensors (light curtains) always consist of an emitter and a receiver. There are diffuse, retro-reflective and through-beam types.

- **Ultrasonic sensors**

Ultrasonic sensors send out a sound pulse in the inaudible range. The echo from the object is processed.



Fig 1.5 Model of Inductive sensor



Fig 1.6 Model of Capacitive sensor



Fig 1.7 Model of Photoelectric sensor



Fig 1.8 Model of Ultrasonic sensor



1.6.4 Classification of sensors based on Functionality

Numerous categories can be used to group sensors. To categorize them as either active or passive is a typical strategy. An active sensor needs an outside power source to react to environmental input and produce output. For instance, to provide meteorological information on the Earth's atmosphere, sensors used in weather satellites frequently need some form of energy. On the other hand, a passive sensor does not require an external power source to pick up environmental input. It draws its energy from the environment, using things like thermal energy or light. The mercury-based glass thermometer is an excellent example. Temperature changes allow the mercury to expand and contract, affecting the level in the glass.

Some sensor types, including seismic and infrared light sensors, come in both active and passive varieties. Which sort of sensor is ideal for the application often depends on the environment in which it is deployed. Some sensor types, including seismic and infrared light sensors, come in both active and passive varieties. The appropriate sort of sensor for an application is often determined by the environment in which it is installed. Depending on the kind of output the sensors provide, analogness or digitalness can also be used to categorise sensors. Environmental input is transformed into output analogue signals using analogue sensors, which are continuous and variable. An illustration would be the thermocouples found in gas hot water heaters.

Based on the kind of output the sensors provide, another method to categorise sensors is by whether they are analogue or digital. Environmental input is transformed into output analogue signals, which are continuous and variable, via analogue sensors. An excellent illustration of an analogue sensor is the thermocouple, which is used in gas hot water heaters. The thermocouple is continuously heated by the pilot light of the water heater. The thermocouple cools if the pilot light goes out, providing a separate analogue signal that says the gas should be turned off.

Unlike analogue sensors, which transform environmental information into discrete digital signals that are delivered in a binary format, digital sensors convert the environment. (1s and 0s). All industries now use a lot of digital sensors, which have mostly replaced analogue ones.

Sensors are also commonly categorized by the type of environmental factors they monitor. Here are some common examples



- **Accelerometer.** This type of sensor detects changes in gravitational acceleration, making it possible to measure tilt, vibration and, of course, acceleration sensors are used in a wide range of industries, from consumer electronics to professional sports to aerospace and aviation.
- **Chemical.** Chemical sensors identify a particular chemical within a medium. (gas, liquid or solid). Chemical sensors can be used to measure a variety of things, including soil nutrient levels in crop fields, smoke or carbon monoxide levels in enclosed spaces, pH levels in bodies of water, the presence of alcohol on someone's breath, and many other things. For instance, the gasoline-to-oxygen ratio will be monitored by an oxygen sensor in a car's emission control system, typically through a chemical reaction that produces voltage. When the mixture is not optimal, a computer in the engine compartment reads the voltage and changes the ratio.
- **Humidity.** These sensors have the ability to measure the amount of water vapour in the air and calculate relative humidity. Since relative humidity is influenced by air temperature, humidity sensors frequently provide temperature values as well. The sensors are utilised in numerous contexts and industries, including as data centers, meteorology, agriculture, manufacturing, and HVAC (heating, ventilation, and air conditioning). (HVAC).
- **Level.** The level of a physical substance, such as water, gasoline, coolant, grain, fertilizer, or garbage, can be determined using a level sensor. For instance, drivers depend on their gas level sensors to prevent being stranded on the side of the road. Systems for warning of tsunamis also employ level sensors.
- **Motion.** Motion detectors can be used to control lights, cameras, parking gates, water faucets, security systems, automated door openers, and a variety of other systems. They can detect physical movement in a specified area (the field of detection). The sensors usually emit some form of energy, such as microwaves, ultrasonic waves, or laser beams, and are able to detect when the energy flow is obstructed by an object that moves in its path.
- **Optical.** Optical sensors, also known as photosensors, are capable of detecting light waves at many wavelengths, including ultraviolet, visible, and infrared light. Optical sensors are widely utilised in a variety of systems, including smartphones, robotics, Blu-ray players, home security systems, and medical equipment.



- **Pressure.** These sensors are widely employed in machinery, vehicles, airplanes, HVAC systems, and other situations to measure the pressure of a liquid or gas. They are crucial in meteorology because they measure atmospheric pressure. Pressure sensors can also be used to monitor the flow of gases or liquids, frequently in order to control the flow.
- **Proximity.** The presence of an object or the distance between objects is detected using proximity sensors. In addition to many other situations, proximity monitors are employed in elevators, assembly lines, parking lots, retail establishments, cars, and robotics.
- **Temperature.** These sensors are capable of measuring the temperature of any target media, including air, liquid, and gas. Appliances, machinery, aircraft, autos, computers, greenhouses, farms, thermostats, and a broad variety of other objects and surroundings all employ temperature sensors.

1.6.5 Advantages of Sensors

Deploying sensors and sensing technology has multiple benefits, including predictive and preventive maintenance. They not only ensure that measurement data is transmitted faster, but also increase accuracy, thereby improving process control, and enhancing asset health. A new breed of sensors are capable of wired and wireless transmission, providing a real-time, continuous data feed from assets and processes. This empowers executives with a more holistic view of a process plant. Businesses that leverage sensors are more connected, secure, and agile than ever before. Improved data capture sensitivity, nearly lossless transmission, and ongoing, real-time analysis are some of sensors' main benefits. Processes are active and are carried out as efficiently as possible thanks to real-time feedback and data analytics services.

The development of sensing technology over time has led to the smart and intelligent sensors of today. Smart sensors have electrical circuits, allowing them to take measurements and output values as digital data, in contrast to standard analogue sensors, which lack active components. These sensors incorporate embedded microprocessors and a signal converter on top of a variety of sensing components. The ability to self-test, self-validate, self-adapt, and self-identify are just a few of the innately intelligent tasks that intelligent sensors are capable of performing. They can recognise conditions, manage a variety of conditions, and comprehend process needs.

1.7 Introduction to Visual Studio

Microsoft's Visual Studio Code, sometimes known as VS Code, is a source-code editor for Windows, Linux, and macOS that uses the Electron Framework. Debugging support, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git are among the features. Users can modify the theme, keyboard shortcuts, preferences, and add functionality by installing extensions.

The most widely used development environment tool among 71,010 respondents in the Stack Overflow 2022 development Survey was Visual Studio Code, with 74.48% saying they use it.



Fig 1.9 Visual Studio and Visual Studio code

The word “radar” is an acronym that stands for Radio Detection and Ranging. This technology has been in use since before World War II to track the speed, position, and direction of ships and aircraft.

Radar systems emit radio waves, a type of electromagnetic energy that can be sent into the air. These signals travel at the speed of light. These waves reflect back from objects in the radar’s path and are processed by the radar receiver. These returned pulses enable the radar to determine the *distance and position* of objects, relative to the radar system’s position.

1.7.1 Features of Visual Studio code

A source-code editor called Visual Studio Code works with many different programming languages, such as C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, and Rust. Based on the Electron framework, which is used to create Node.js web apps that employ the Blink layout engine, it is used to create mobile applications. The editor component (codenamed "Monaco") used in Azure DevOps is also utilised in Visual Studio Code.



(formerly called Visual Studio Online and Visual Studio Team Services). The majority of popular programming languages have minimal support out of the box in Visual Studio Code. This fundamental support consists of configurable snippets, code folding, bracket matching, and syntax highlighting. Along with debugging support for Node.js, Visual Studio Code also comes with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML. It enables users to open one or more directories, which may subsequently be saved in workspaces for later usage in place of a project system. As a result, it can function as a language-neutral code editor for any language. It supports a wide variety of programming languages, each with its own set of capabilities. Through the settings, unwanted files and folders can be excluded from the project tree.

Many aspects of Visual Studio Code can be accessible using the command palette instead of through menus or the user interface. Extensions are add-ons for Visual Studio Code that are accessible through a single repository. This contains language support and editor additions. The capacity to design extensions that add support for new languages, themes, and other features is a noteworthy feature. Visual Studio Code comes with a built-in tool for source control. Users can access version control settings and see changes made to the current project by using a dedicated tab in the menu bar. Any compatible version control system must be linked to Visual Studio Code in order to use the feature. (Git, Apache Subversion, Perforce, etc.). This enables users to push and pull requests from within the Visual Studio Code software, as well as create repositories. As a result of its numerous FTP extensions, Visual Studio Code can be used as a cost-free substitute for web development. Without having to download any additional software, code may be synced between the editor and the server.

1.8 Introduction to Arduino

Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as **Windows, Mac OS X, and Linux**. It supports the programming languages C and C++. Here, IDE stands for **Integrated Development Environment**.

The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

The Arduino IDE will appear as:

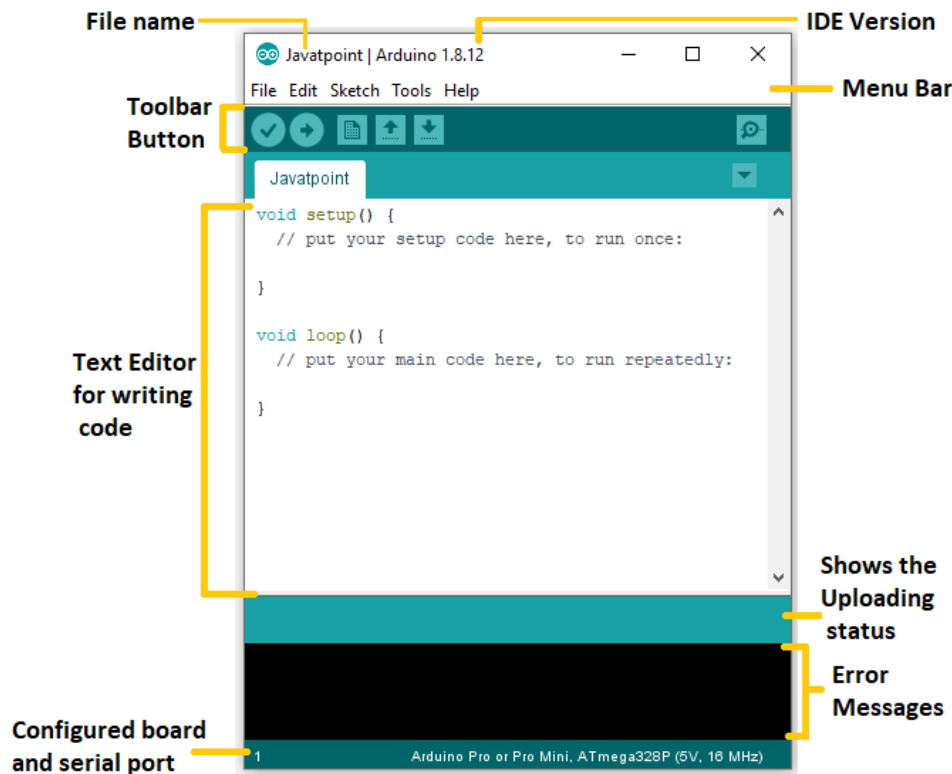


Fig 1.10 Arduino IDE Interface

Extensions are add-ons for Visual Studio Code that are accessible through a single repository. This contains language support and editor additions. The capacity to design extensions that add support for new languages, themes, and other features is a noteworthy feature.

1.8.1 Arduino UNO Board

A microcontroller board called Arduino UNO is based on the ATmega328P. It contains 6 analogue inputs, a 16 MHz ceramic resonator, 14 digital input/output pins (six of which can be used as PWM outputs), a USB port, a power jack, an ICSP header, and a reset button. It comes with everything needed to support the microcontroller; to get started, just plug in a USB cable, an AC-to-DC adapter, or a battery. You can experiment with your UNO without being overly concerned that you'll make a mistake; in the worst case, you can replace the chip for a few dollars and start over.

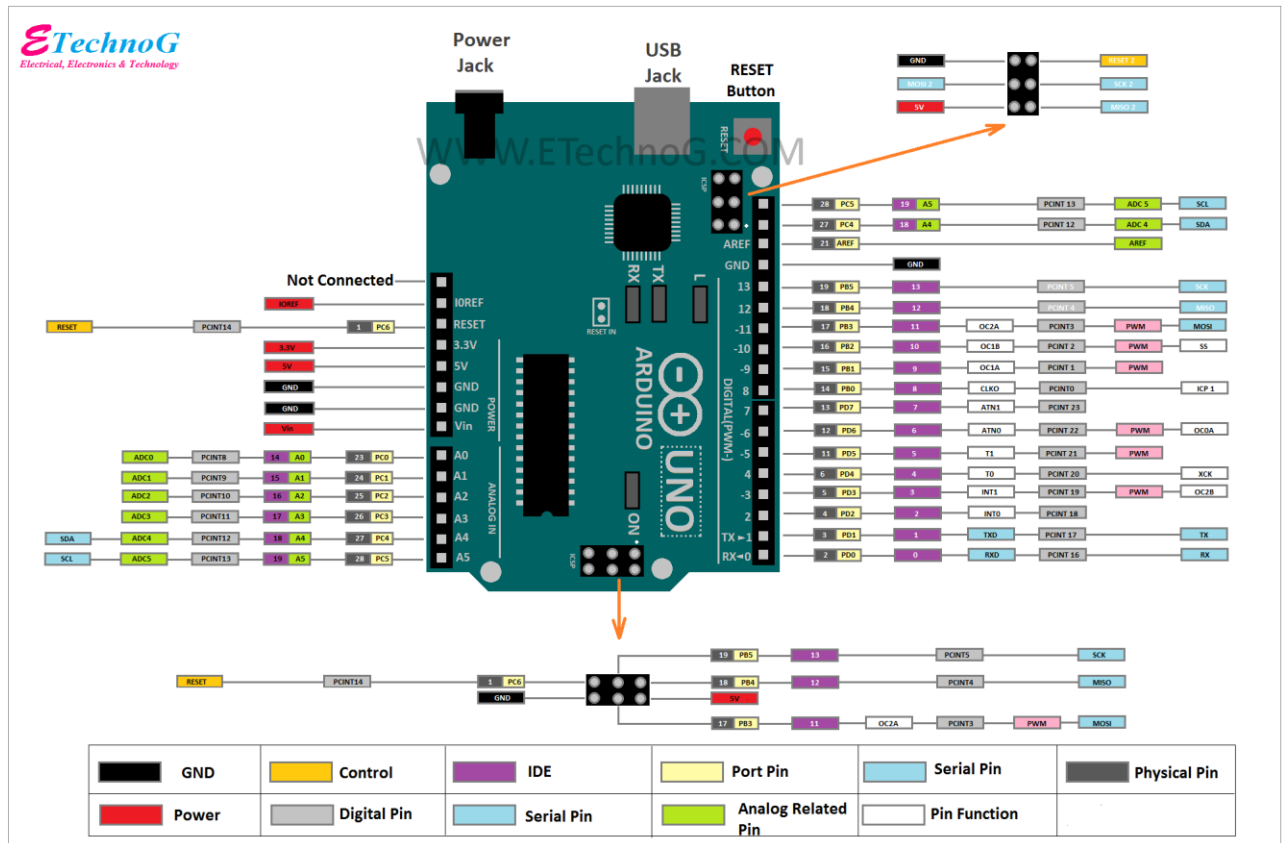


Fig 1.11 Arduino UNO Pinout Diagram

This is a description of what you will see when you look at the board from the top down; areas of the board that you might interact with during regular use are highlighted.

In a clockwise direction from the top centre:

- Orange Analog Reference pin
- (Light Green) Digital Ground
- Green digital pins 2–13
- Dark green Digital Pins 0-1/Serial In/Out - TX/RX If serial communication is also being used (for example, via `Serial.begin()`), then these pins cannot be used for digital input and output (`digitalRead` and `digitalWrite`).
- Reset Button - S1 (dark blue)
- Blue-green In-Circuit Serial Programmer
- Pins 0–5 for analogue in (light blue)
- Ground and power pins (grounds: light orange, power: orange)
- (9–12 VDC) External Power Supply In - X1 (pink)



- Toggles Place the jumper on the two pins that are closest to the selected source for external and USB power. - Purple SV1
- USB (used to power the board, to upload sketches to the board, and for serial communication between the board and the computer) (yellow)

1.8.2 Microcontroller

ATmega328P microcontrollers (found on modern circuit boards)

14 digital I/O pins, 6 of which are used to produce PWM.

Pins for analogue input: 6 (DIP) or 8 (SMD).

DC I/O pin current: 40 mA

32 KB Flash Memory; 2 KB SRAM

(Used on the majority of Arduino Diecimila and early Duemilanove) EEPROM: 1 KB

ATmega168

14 digital I/O pins, 6 of which are used to produce PWM.

Pins for analogue input: 6 (DIP) or 8 (SMD).

DC I/O pin current: 40 mA

16 KB Flash Memory

1 KB SRAM

512-byte

ATmega8 EEPROM (used on some older boards)

14 digital I/O pins, 3 of which are used to produce PWM.

6 DC Analog Input Pins I/O pin current: 40 mA

8 KB of flash memory

1 KB SRAM

1.9 Java Script

Along with HTML and CSS, the programming language JavaScript, sometimes known as JS, is one of the foundational elements of the World Wide Web. 98% of websites will utilize JavaScript on the client side by 2022 to control webpage behaviour, frequently integrating third-party libraries. A dedicated JavaScript engine is available in every major web browser and is used to run the code on users' devices.

JavaScript is an ECMAScript-compliant high-level, frequently just-in-time compiled language. It features first-class functions, prototype-based object orientation, and dynamic typing. It supports event-driven, functional, and imperative programming paradigms and is a multi-paradigm. For working with text, dates, regular expressions, shared data structures, and the Document Object Model (DOM), it includes application programming interfaces (APIs).



Fig 1.12 Java script Reference

There are no input/output (I/O) features like networking, storage, or graphics capabilities in the ECMAScript standard. In reality, JavaScript I/O APIs are offered by the web browser or another runtime system. Originally only used in web browsers, JavaScript engines are now essential parts of some servers and a wide range of applications. Node.js is the most widely used runtime system for this application. Even while Java and JavaScript share the same name, syntax, and standard libraries, the two programming languages are separate and have very different designs.

For their client-side scripting, more than 80% of websites employ a third-party JavaScript library or web framework. Over 75% of websites utilize jQuery, making it by far the most popular client-side library. Twitter currently uses the React library, which was developed by Facebook for its website and then made available as open source. Similar to this, Google's Angular framework, which it developed for its websites like YouTube and Gmail, is now an open-source project that other people utilize. On the other hand, websites that solely rely on ordinary JavaScript capabilities and don't use any libraries or frameworks are referred to as "Vanilla JS" websites.



1.9.1 Features of Java Script

Structured and organized

Most of the structured programming syntax from C, including if statements, while loops, switch statements, do while loops, etc., is supported by JavaScript. Scoping is one example of a partial exception; prior to ECMAScript 2015, block scoping was only available in JavaScript and was added with the `let` and `const` keywords. Expressions and statements are distinct in JavaScript, as they are in C. Automatic semicolon insertion, which allows semicolons (which terminate statements) to be omitted, is one syntactic distinction from C.

Weakly Typed

Because JavaScript is loosely typed, depending on the operation employed, some types are implicitly cast.

Values are cast to strings like the following:

- Strings are left as-is
- Numbers are converted to their string representation
- Arrays have their elements cast to strings after which they are joined by commas
(,)
- Other objects are converted to the string `[object Object]` where `Object` is the name of the constructor of the object

Values are cast to numbers by casting to strings and then casting the strings to numbers. These processes can be modified by defining `to string` and `value Of` functions on the prototype for string and number casting respectively.

JavaScript has received criticism for the way it implements these conversions as the complexity of the rules can be mistaken for inconsistency. For example, when adding a number to a string, the number will be cast to a string before performing concatenation, but when subtracting a number from a string, the string is cast to a number before performing subtraction.

Typing

JavaScript is dynamically typed like most other scripting languages. A type is associated with a value rather than an expression. For example, a variable initially bound to a number may be reassigned to a string. JavaScript supports various ways to test the type of objects, including duck typing.



Run-time evaluation

JavaScript includes an `eval` function that can execute statements provided as strings at run-time.

Object-orientation (prototype-based)

Prototypal inheritance in JavaScript is described by Douglas Crockford as:

You make prototype objects, and then ... make new instances. Objects are mutable in JavaScript, so we can augment the new instances, giving them new fields and methods. These can then act as prototypes for even newer objects. We don't need classes to make lots of similar objects... Objects inherit from objects.

In JavaScript, an object is an associative array, augmented with a prototype (see below); each key provides the name for an object property, and there are two syntactical ways to specify such a name: dot notation (`obj.x = 10`) and bracket notation (`obj['x'] = 10`). A property may be added, rebound, or deleted at run-time. Most properties of an object (and any property that belongs to an object's prototype inheritance chain) can be enumerated using a `for...in` loop.

1.10 Introduction to Three.js

Three.js is a JavaScript-based WebGL engine that enables browser-based GPU-powered gaming and other graphics-intensive applications. Many capabilities and APIs are available in the three.js library for generating 3D scenes in your browser. Without the use of specialized browser plugins, Three.js enables the development of 3D animations that are GPU-accelerated and integrated into websites using JavaScript. The development of WebGL, a low-level graphics API designed exclusively for the web, has made this possible. Complex 3D computer animations can be created for browser display without the work needed for a traditional standalone application or a plugin thanks to high-level tools like Three.js or GLGE, SceneJS, PhiloGL, and many others.

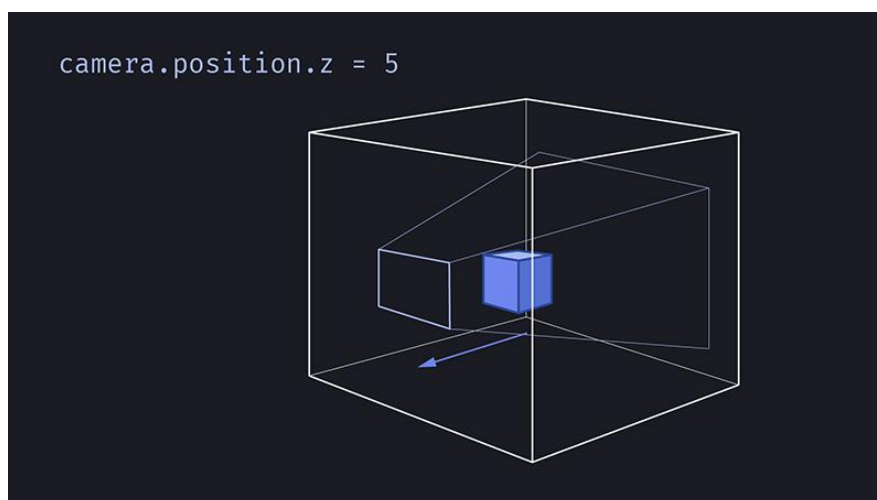


Fig 1.13 Object in 3d space



1.11 Introduction to Fusion 360

Fusion 360 is a for-profit computer-aided engineering (CAE), computer-aided manufacturing (CAM), and printed circuit board (PCB) design software program created by Autodesk. It is accessible on Windows, macOS, and web browsers and has streamlined apps for Android and iOS. Fusion 360 has a limited home-based, non-commercial personal edition that is available for free but requires a paid membership.

1.11.1 History of fusion 360.

On September 24, 2013, Autodesk unveiled Fusion 360. It replaced Inventor Fusion, which it replaced with many of its capabilities. After its release, Fusion 360 included the following additional Autodesk products:

The Autodesk 123D Slicer feature was integrated in 2017.

Autodesk Meshmixer was terminated in 2021 when Fusion 360 merged its capability.

Fusion 360 and Netfabb were combined in 2021.

1.11.2 Features of Fusion 360.

Sheet metal, simulation, documentation, and 3D modelling are all included in Fusion 360. It can control production procedures including milling, turning, machining, and additive manufacturing. Additionally, it provides electrical design automation (EDA) functions including component management, PCB design, and schema design. Additionally, it can be used for a variety of complex simulation jobs (FEA), rendering, animation, and generative design.

List of main features are:

- 3D Design & Modelling
- Manufacturing (CAM)
- Data Management
- Electronics (EDA, PCB)
- Team Collaboration
- Generative Design
- Additive Manufacturing (3D Printing)
- Simulation (FEA)
- Documentation & Drawings

Building a digital twin of a screw jack for predictive maintenance

Fusion 360 provides a variety of design areas with tools including workspaces for modelling, simulating, manufacturing, sheet metal, surfacing, and meshing. You can utilise a variety of expert tools after choosing the right workplace to produce simple forms for brainstorming.

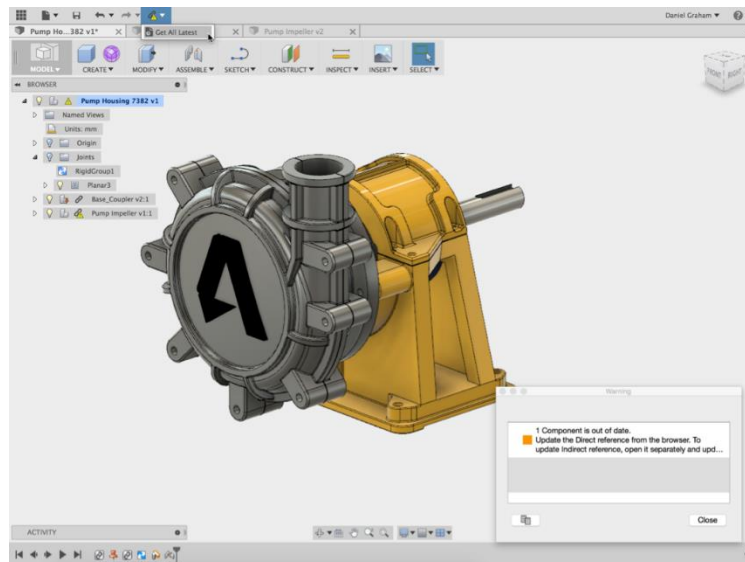


Fig 1.14(a) Fusion 360 (Model 1)

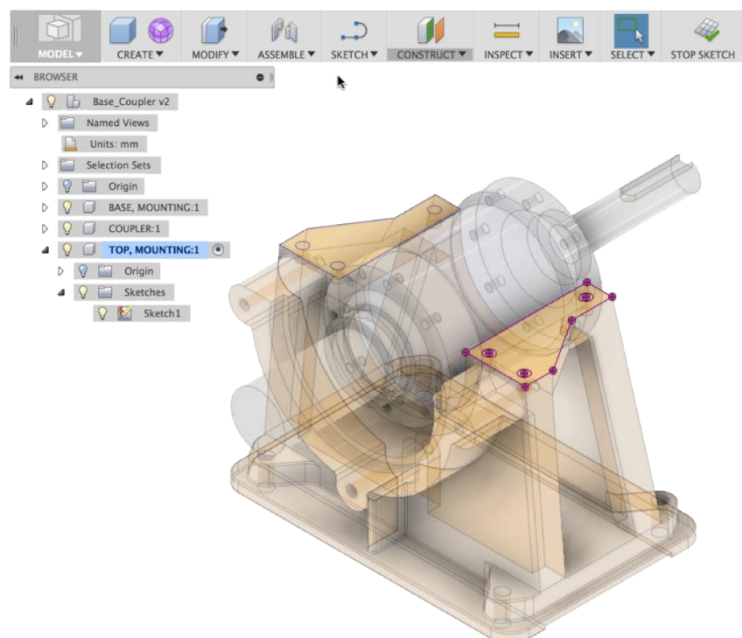


Fig 1.14(b) Fusion 360 (Model 2)



1.12 Screw Jack

A jackscrew, often known as a screw jack, is a jack that is controlled by turning a leadscrew. It is frequently used as an adjustable support for high loads, such as the foundations of houses, as well as to lift moderately and heavily loaded objects, such as cars, and to raise and lower the horizontal stabilizers of airplanes. A screw jack is a fixed support structure with a large, flat base that is sitting on the ground and a heavy-duty vertical screw with a load table installed on its top. The handle, a metal bar, slots into holes in a rotating collar on the screw's head. The load lying on the load table is raised when the handle is cranked clockwise because the screw emerges from the base further. The screw is often made with Acme threads to accommodate heavy load forces.

1.12.1 Types of screw jacks

There are 3 main types of screw jacks:

- Machine/worm gear screw jacks
- Ball screw jacks
- Bevel gear jacks.

1.12.1.1 Machine/worm gear screw jack

Mechanical worm gear screw jack offered with lifting nut or translating lifting screw in addition to trapezoidal lifting screw. Both upright and inverted configurations are offered. Up to 200 tonnes of capacity.

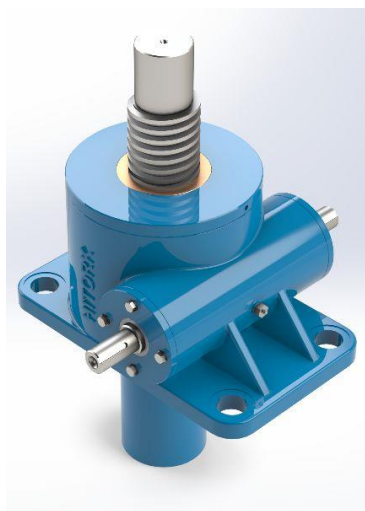


Fig 1.15(a) Machine/worm gear screw jack

1.12.1.2 Ball screw jack

Ball screw jacks use ball nuts that are roughly 90% as effective as machine screw jacks, therefore they use much less energy to lift an equivalent load.



Fig 1.15(b) Ball screw jack

1.12.1.3 Bevel screw jack

Bevel Gear Ball Screw Jacks shift the load smoothly and effectively by using a ball screw and nut made of hardened alloy steel that carry the load between the nut and the screw.



Fig 1.15(c) Bevel screw jack

1.12.2 Advantages of screw jack

In comparison to some other types of jacks, jackscrews have the benefit of being self-locking, which means that no matter how much weight is being supported, the screw will remain stationary where it was left and won't rotate backwards. Compared to hydraulic jacks, for



instance, which will travel rearward when loaded if the force on the hydraulic actuator is unintentionally withdrawn, they are hence fundamentally safer.

1.12.2.1 Mechanical Advantage

The ideal mechanical advantage of a screw jack, the ratio of the force the jack exerts on the load to the input force on the lever ignoring friction is

$$\frac{F_{\text{load}}}{F_{\text{in}}} = \frac{2\pi r}{l}$$

Where

F_{load} is the force the jack exerts on the load.

F_{in} is the rotational force exerted on the handle of the jack

r is the length of the jack handle, from the screw axis to where the force is applied

l is the lead of the screw.

The screw jack consists of two simple machines in series; the long operating handle serves as a lever whose output force turns the screw. So, the mechanical advantage is increased by a longer handle as well as a finer screw thread. However, most screw jacks have large amounts of friction which increase the input force necessary, so the actual mechanical advantage is often only 30% to 50%.

1.12.3 Limitations

Screw jacks can only raise so much weight. The friction in the screw threads increases as the load increases. A fine pitch thread decreases the speed at which the jack can operate while increasing the advantage of the screw. The point at which a longer functioning lever simply bends at its inner end is quickly reached.

Hydraulic jacks have now largely taken the place of screw jacks. This was supported in 1858 when, following two unsuccessful attempts by other means, Brunel's SS Great Eastern was successfully launched using jacks by the Tangye business in accordance with Bramah's hydraulic press design. The highest mechanical advantage that a hydraulic jack can provide is not constrained by the restrictions placed on screw jacks and may even be much higher.

1.13 Load cell

A load cell transforms a force into a measurable electrical output. The electrical signal varies in direct proportion to the applied force. Strain gauges, pneumatic, and hydraulic load cells are among the several varieties. We'll talk about strain gauge load cells in this tutorial.

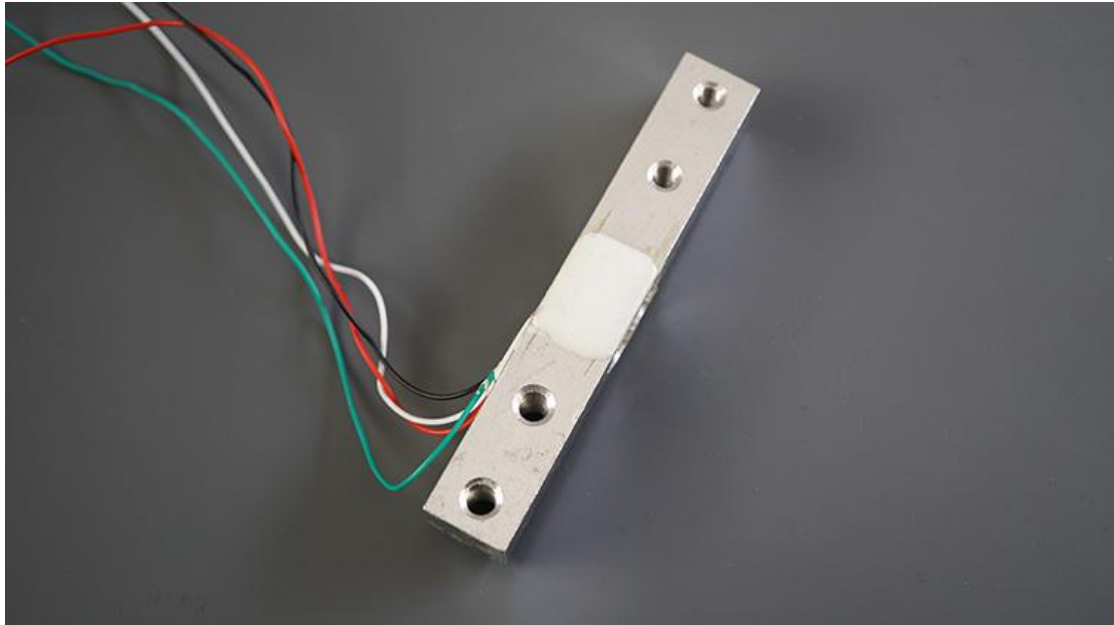


Fig 1.16 Load cell

A metal bar with mounted strain gauges (behind the white glue in the image above) makes up strain gauge load cells. An electrical sensor called a strain gauge monitors the force or strain on an item. When an object experiences an external force that causes the object's shape (in this case, the metal bar) to change, the resistance of the strain gauges changes. We can determine an object's weight because the resistance changes proportionally to the applied load.

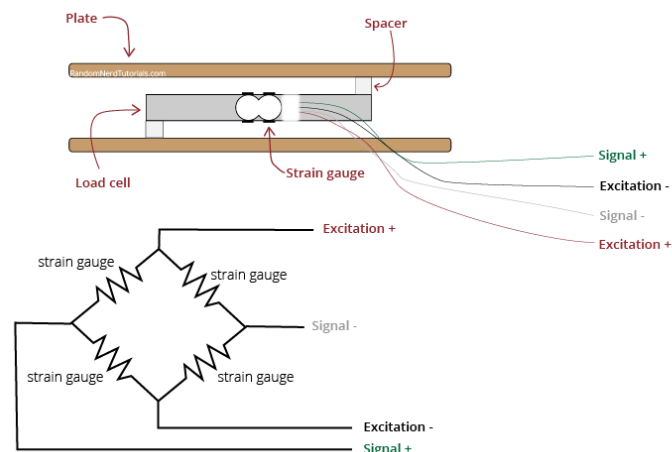


Fig 1.17 Wheatstone bridge

1.14 Magnetic encoder

The magnetic encoder recognises variations in the magnetic field as information about rotational position, transforms those changes into electrical signals, and outputs those signals. A permanent magnet plus a magnetic sensor make up the most basic magnetic encoder.



Fig 1.18 Magnetic encoder

1.15 Vue Frame work

Vue is a JavaScript framework for creating user interfaces, and it is pronounced similarly to view (/vju/). It provides a declarative and component-based programming model that enables you to effectively create user interfaces, whether they are straightforward or intricate. It builds on top of common HTML, CSS, and JavaScript.

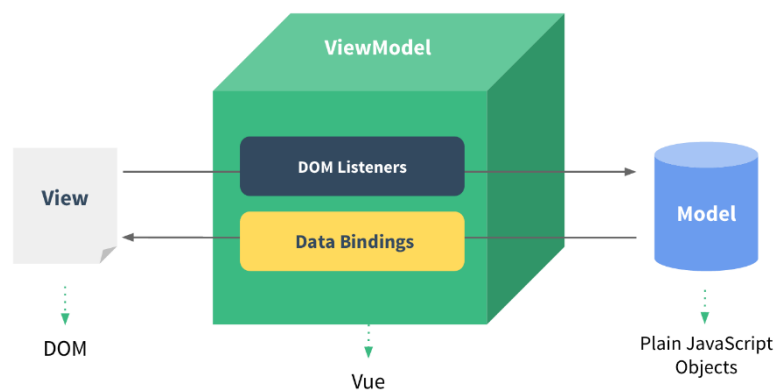


Fig 1.19 Architecture of Vue



1.16 Electron-JavaScript Library

The OpenJS Foundation created and maintains the free and open-source software framework known as Electron (formerly Atom Shell). The framework is made to enable the development of desktop applications that make use of web technologies (primarily HTML, CSS, and JavaScript, though other technologies like front-end frameworks and Web Assembly are also feasible), which are then rendered using a variant of the Chromium browser engine and have a back end that makes use of the Node.js runtime environment. Additionally, it makes use of a number of APIs to enable features like a module for inter-process communication and native integration with Node.js services.

The primary GUI foundation for several open-source applications, including Atom, GitHub Desktop, Light Table, Visual Studio Code, WordPress Desktop and Eclipse Theia, is Electron, which was initially developed for Atom

1.16.1 Architecture of Electron

A "main" process and a number of "renderer" processes are both present in electron applications. By creating an instance of the Browser Window class, the main process can create other renderer processes after running the application's logic (such as menus, shell commands, and lifecycle events). The Browser Window class loads a window that appears on the screen by rendering HTML and CSS.

If the node Integration field in the main process is set to true, both the renderer and main processes can function with Node.js integration. The majority of Electron's APIs are built in C++ or Objective-C and made immediately accessible to the application code via JavaScript bindings.



CHAPTER 2

LITERATURE REVIEW



1. Literature Review

1.1. Digital twin technology

- **Michael G. Kapteyn, Karen E. Willcox : Design of Digital Twin Sensing Strategies Via Predictive Modeling and Interpretable Machine Learning**

The methodology for sensor placement and dynamic sensor scheduling decisions for digital twins is developed in this work. Predictive models are used to train the best classification trees that represent the map from observed data to estimated digital twin states. The assimilation of digital twin data is framed as a classification problem. The resulting classification trees produce an understandable mathematical representation that can be queried to help with sensor location and sensor scheduling decisions, in addition to giving a rapid digital twin updating capability. The suggested method is illustrated using a structural digital twin of an unmanned aerial vehicle with a 12 foot wingspan. By modelling situations with predictive reduced-order models of the vehicle in a variety of structural states, training data are produced offline.

- **Xiaonan Lai, Xiwang He, Shuo Wang : Building a Lightweight Digital Twin of a Crane Boom for Structural Safety Monitoring Based on a Multifidelity Surrogate Model**

This study analyzed the Unrecognized wear and tear from overloading and tiredness at critical locations is one of the major dangers in construction that might cause structural failure in the crane boom. Consequently, the stability of the crane boom To make sure the crane boom is operating within the intended load capacity, it should be continuously monitored. To increase the real-time monitoring and prediction accuracy of the structural safety of a crane boom, we created a lightweight digital twin using the multifidelity surrogate (MFS) model in this work. The use of digital twin technology for online monitoring and analysis of buildings, machinery, and even human bodies has great promise because it allows for real-time mapping between the physical and digital worlds. Using sensor data and the MFS model, the lightweight digital twin may dynamically mirror the crane boom postures and predict its structural performance in real time.

- **Xiaonan Lai, Xiwang He, Yong Pang : A Scalable Digital Twin Framework Based on a Novel Adaptive Ensemble Surrogate Model**

This work deals with ensembling offers reliable approximation in addition to reduces the number of ISMs participating in the ensemble by multicriterion model screening, which also lowers the additional cost brought on by the ensemble. Moreover, A node



rearrangement approach that offers scalability for the creation of a digital model is proposed and is based on the features of the finite element method. In other words, the distribution and number of nodes can be altered to acquire the information at strategic locations while simultaneously lowering the computational cost by reducing the number of nodes. The effectiveness of the suggested ensemble and node rearrangement strategy is tested using numerical experiments. To demonstrate the viability of creating scalable digital twins, a telehandler is utilised as an example.

- **Ricard Sanchis, Salvador Cardona, Jordi Martinez : Determination of the Vertical Vibration of a Ballasted Railway Track to Be Used in the Experimental Detection of Wheel Flats in Metropolitan Railways**

This paper presents a mathematical model used to obtain the vertical vibration of a ballasted railway track when a wheel is passing at a certain speed over a fixed location of the rail. The aim of this simulation is to compare calculated root-mean-square (RMS) values of the vertical vibration velocity with measured RMS values. This comparison is the basis for a proposed time domain methodology for detecting potential wheel flats or any other singular defect on the wheel rolling bands of metropolitan trains. In order to reach this goal, a wheel–rail contact model is proposed; this model is described by the track vertical impulse response and the vertical impulse response of the wheel with the primary suspension, both linked through a Hertz nonlinear stiffness.

- **Aidan Fuller, , Zhong Fan, Charles Day : Digital Twin: Enabling Technologies, Challenges and Open Research**

The paper provides an assessment of the enabling technologies, challenges and open research for Digital Twins. There are many definitions of the Digital Twin, but the best described as the seamless integration of data going either way between a physical and virtual machine. The problems, uses, and technology that makes artificial intelligence and the Internet of Things (IoT) possible as well as Digital Twins are shown. A categorical review of recent studies is produced after a review of publications on digital twins. Manufacturing, healthcare, and smart cities have been classed by study topics, and the review discusses a variety of publications that reflect these areas and the current level of research in each of these fields.



1.2. Three.js

- **Brian Danchilla: Three.js Framework, Springer, 2012**

In this chapter we will concentrate on one of the most widely used frameworks — Three.js. We will cover the following: A background of the library, How to start development with Three.js, Falling back to a 2D canvas context for rendering if WebGL is not supported, Three.js API calls to easily create cameras, objects, and use lighting models, Show the equivalent Three.js code to some examples found in previous chapters, which used direct low-level WebGL API calls

- **Rushikesh Kore, Hansel D'silva, Augmented Reality Clothes Fitting Web App Using Three.js & Posenet**

In this paper we have learned the use of augmented reality, we have investigated various methodologies for simulating the experience of a clothing fitting room. With the help of their webcam, potential clients of fashion e-commerce can virtually try-on any outfit of their choice thanks to a web application we designed. With the help of the PoseNet 2.0 Tensorflow.js package for posture estimation and the three.js library for 3D model animation, our solution divides this challenge into two steps. These procedures have been succinctly explained, along with the appropriate visuals. This research study gives a quick overview of virtual clothing fitting utilising augmented reality technologies.

1.3. Screw Jack

- **Aashish Gyawali, Modeling & Structural Analysis of Screw Jack:**

This study aims to give brief information about the Screw jack and its structural analysis. This project is about drawing components and preparation of assembly models of screw jack.



CHAPTER 3

METHODOLOGY

3. Methodology

There are two ways to deploy digital twin technology in this scenario. They are:

1. Processing applications while utilising an Arduino.
2. Using three.js, Fusion 360 and webapp.

3.1 In Processing application while using an Arduino

In this method an Arduino UNO board, MPU 650 Sensor, processing application and Arduino Application are required.

Step by step procedure:

- Initially Arduino application is to be downloaded in the windows / mac based on the availability.
- MPU 650 Sensor has to be connected with Arduino board using the jumper wires as shown in the below figure.



Fig 3.1 MPU 6050 Sensor

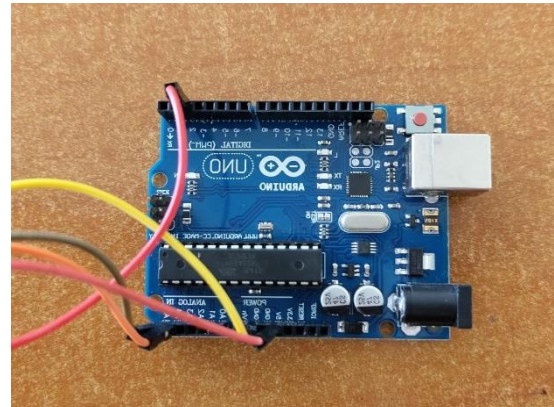


Fig 3.2 Arduino UNO Board

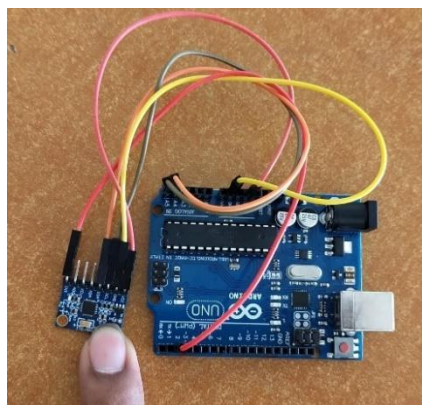


Fig 3.3 Sensor and Arduino Board Integration

- Now the entire Sensor – Arduino fusion is to be connected to the computer using Micro USB Cable A-B as shown in the below figure.

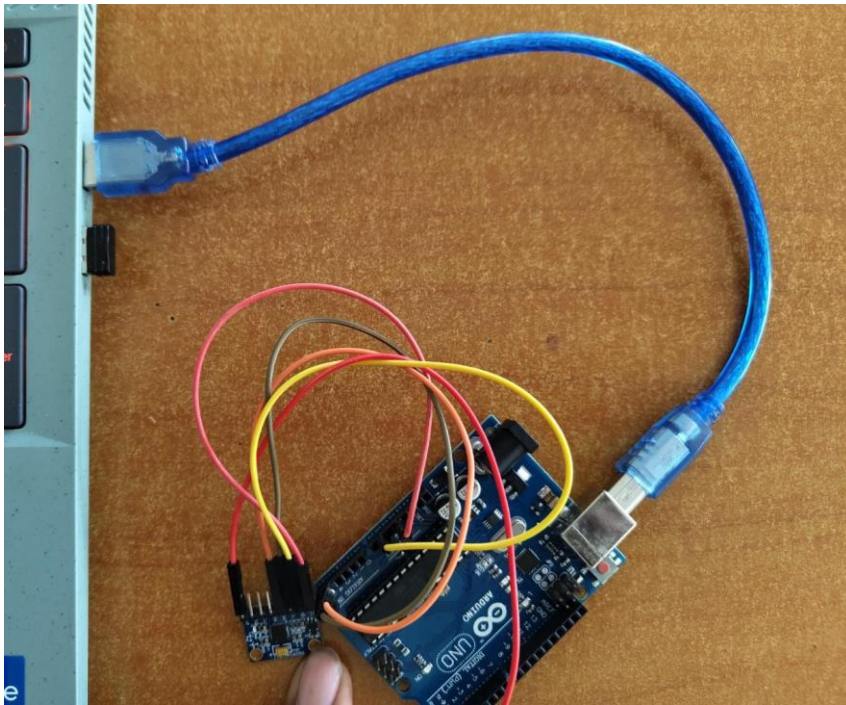
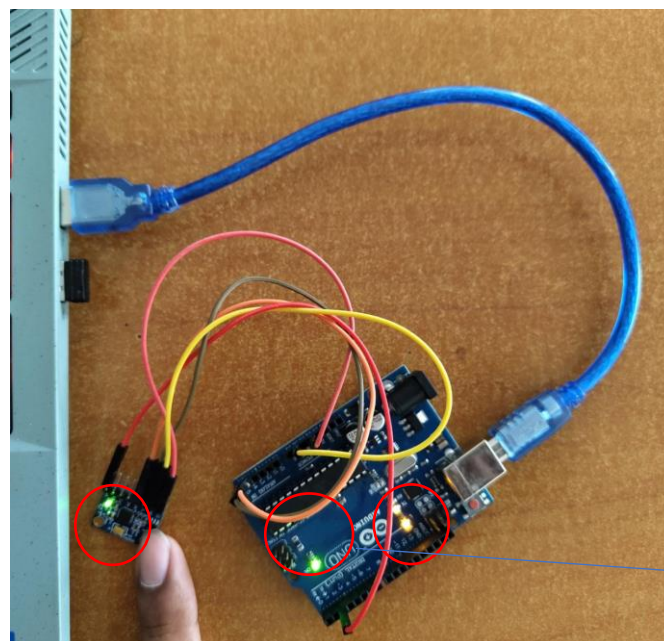


Fig 3.4(a) Integration with the system (Ideal case)

- Make sure that the LED bulbs of sensor and Arduino board are blinking / Switched on as shown in Fig 3.4 so that it is ensured that all the components are in proper working condition.



Lights are
Switched on /
Blinking

Fig 3.4(b) Integration with the system (Connected case)

- Immediately after the setup is done, Arduino Application is to be opened which will look as shown in Fig 3.5

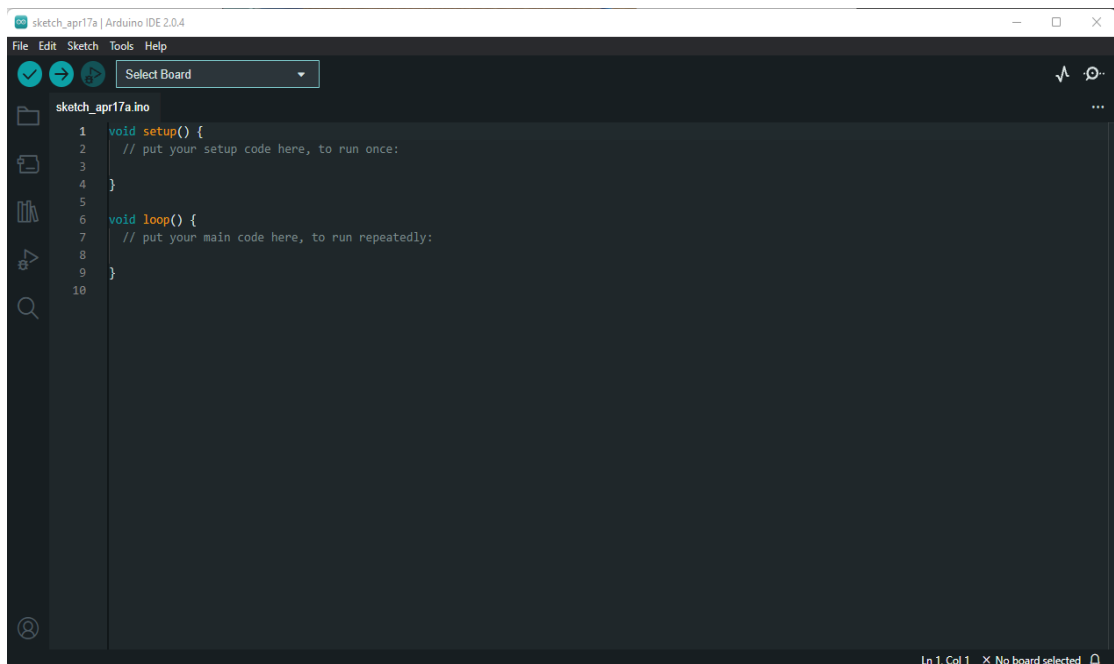


Fig 3.5 Interface of Arduino IDE

- Select the board by clicking on “Select Board” Dropdown. Choose the Required board based on the ports available and port that is mentioned in the code. The available ports are shown in the Fig 3.6

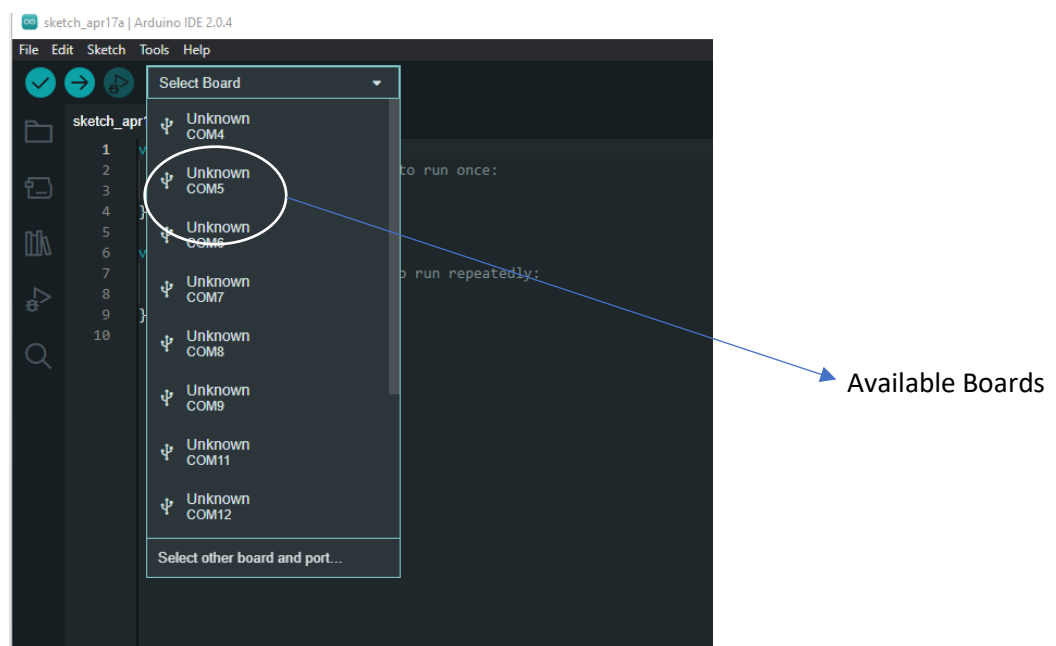


Fig 3.6 Interface of Arduino IDE representing boards



- Based on the Port selected and Sensor configurations the code need to be developed in the Arduino IDE.
- Arduino code developed to integrate MPU 6050 sensor and Arduino UNO Board is given below

```
=====

#include "I2Cdev.h"

#include "MPU6050_6Axis_MotionApps20.h"

// #include "MPU6050.h" // not necessary if using MotionApps include file

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h

// #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
//     #include "Wire.h"
// #endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for SparkFun breakout and InvenSense evaluation board)
// AD0 high = 0x69

MPU6050 mpu;

// MPU6050 mpu(0x69); // <-- use for AD0 high

/* =====

NOTE: In addition to connection 3.3v, GND, SDA, and SCL, this sketch
depends on the MPU-6050's INT pin being connected to the Arduino's
external interrupt #0 pin. On the Arduino Uno and Mega 2560, this is
digital I/O pin 2.

* ===== */

/* =====

NOTE: Arduino v1.0.1 with the Leonardo board generates a compile error
when using Serial.write(buf, len). The Teapot output uses this method.
The solution requires a modification to the Arduino USBAPI.h file, which
is fortunately simple, but annoying. This will be fixed in the next IDE
```



Building a digital twin of a screw jack for predictive maintenance

release. For more info, see these links:

<http://arduino.cc/forum/index.php/topic,109987.0.html>

<http://code.google.com/p/arduino/issues/detail?id=958>

```
* ===== */
```

```
// uncomment "OUTPUT_READABLE_QUATERNION" if you want to see the actual
```

```
// quaternion components in a [w, x, y, z] format (not best for parsing
```

```
// on a remote host such as Processing or something though)
```

```
// #define OUTPUT_READABLE_QUATERNION
```

```
#define OUTPUT_READABLE_YAWPITCHROLL
```

```
// uncomment "OUTPUT_TEAPOT" if you want output that matches the
```

```
// format used for the InvenSense teapot demo
```

```
#define OUTPUT_TEAPOT
```

```
#define INTERRUPT_PIN 2 // use pin 2 on Arduino Uno & most boards
```

```
#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)
```

```
bool blinkState = false;
```

```
// MPU control/status vars
```

```
bool dmpReady = false; // set true if DMP init was successful
```

```
uint8_t mpulntStatus; // holds actual interrupt status byte from MPU
```

```
uint8_t devStatus; // return status after each device operation (0 = success, !0 = error)
```

```
uint16_t packetSize; // expected DMP packet size (default is 42 bytes)
```

```
uint16_t fifoCount; // count of all bytes currently in FIFO
```

```
uint8_t fifoBuffer[64]; // FIFO storage buffer
```




Building a digital twin of a screw jack for predictive maintenance

```
// orientation/motion vars

Quaternion q;      // [w, x, y, z]    quaternion container
VectorInt16 aa;    // [x, y, z]      accel sensor measurements
VectorInt16 aaReal; // [x, y, z]      gravity-free accel sensor measurements
VectorInt16 aaWorld; // [x, y, z]    world-frame accel sensor measurements
VectorFloat gravity; // [x, y, z]    gravity vector
float euler[3];    // [psi, theta, phi] Euler angle container
float ypr[3];      // [yaw, pitch, roll] yaw/pitch/roll container and gravity vector


// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };


// =====
// ===      INTERRUPT DETECTION ROUTINE      ===
// =====

volatile bool mpuInterrupt = false; // indicates whether MPU interrupt pin has gone high
void dmpDataReady() {
    mpuInterrupt = true;
}


// =====
// ===      INITIAL SETUP      ===
// =====

void setup() {
    // join I2C bus (I2Cdev library doesn't do this automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
        Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation difficulties
    #endif
}
```



Building a digital twin of a screw jack for predictive maintenance

```
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE

    Fastwire::setup(400, true);

#endif

Serial.begin(115200);

while (!Serial); // wait for Leonardo enumeration, others continue immediately


// initialize device

Serial.println(F("Initializing I2C devices..."));

mpu.initialize();

pinMode(INTERRUPT_PIN, INPUT);


// verify connection

Serial.println(F("Testing device connections..."));

Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050
connection failed"));


// wait for ready

Serial.println(F("\nSend any character to begin DMP programming and demo: "));

while (Serial.available() && Serial.read()); // empty buffer

while (!Serial.available());           // wait for data

while (Serial.available() && Serial.read()); // empty buffer again


// load and configure the DMP

Serial.println(F("Initializing DMP..."));

devStatus = mpu.dmpInitialize();


// supply your own gyro offsets here, scaled for min sensitivity

mpu.setXGyroOffset(220);
mpu.setYGyroOffset(76);
```



Building a digital twin of a screw jack for predictive maintenance

```
mpu.setZGyroOffset(-85);

mpu.setZAccelOffset(1788); // 1688 factory default for my test chip

// make sure it worked (returns 0 if so)
if (devStatus == 0) {
    // Calibration Time: generate offsets and calibrate our MPU6050
    mpu.CalibrateAccel(6);
    mpu.CalibrateGyro(6);
    mpu.PrintActiveOffsets();
    // turn on the DMP, now that it's ready
    Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);

    // enable Arduino interrupt detection
    Serial.print(F("Enabling interrupt detection (Arduino external interrupt "));
    Serial.print(digitalPinToInterrupt(INTERRUPT_PIN));
    Serial.println(F(")..."));
    attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();

    // set our DMP Ready flag so the main loop() function knows it's okay to use it
    Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;

    // get expected DMP packet size for later comparison
    packetSize = mpu.dmpGetFIFOPageSize();
} else {
    Serial.print(F("DMP Initialization failed (code "));
    Serial.print(devStatus);
    Serial.println(F(")"));
}
```



```
// configure LED for output

pinMode(LED_PIN, OUTPUT);
}

// =====
// ===      MAIN PROGRAM LOOP      ===
// =====

void loop() {
    // if programming failed, don't try to do anything
    if (!dmpReady) return;
    // read a packet from FIFO
    if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) { // Get the Latest packet
        #ifdef OUTPUT_READABLE_QUATERNION
            // display quaternion values in easy matrix form: w x y z
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            Serial.print("quat\t");
            Serial.print(q.w);
            Serial.print("\t");
            Serial.print(q.x);
            Serial.print("\t");
            Serial.print(q.y);
            Serial.print("\t");
            Serial.println(q.z);
        #endif
    }
}
```



```
#ifdef OUTPUT_READABLE_EULER

// display Euler angles in degrees

mpu.dmpGetQuaternion(&q, fifoBuffer);

mpu.dmpGetEuler(euler, &q);

Serial.print("euler\t");

Serial.print(euler[0] * 180/M_PI);

Serial.print("\t");

Serial.print(euler[1] * 180/M_PI);

Serial.print("\t");

Serial.println(euler[2] * 180/M_PI);

#endif

#ifdef OUTPUT_READABLE_YAWPITCHROLL

// display Euler angles in degrees

mpu.dmpGetQuaternion(&q, fifoBuffer);

mpu.dmpGetGravity(&gravity, &q);

mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

Serial.print("ypr\t");

Serial.print(ypr[0] * 180/M_PI);

Serial.print("\t");

Serial.print(ypr[1] * 180/M_PI);

Serial.print("\t");

Serial.println(ypr[2] * 180/M_PI);

#endif

#ifdef OUTPUT_READABLE_REALACCEL

// display real acceleration, adjusted to remove gravity

mpu.dmpGetQuaternion(&q, fifoBuffer);

mpu.dmpGetAccel(&aa, fifoBuffer);

mpu.dmpGetGravity(&gravity, &q);
```



Building a digital twin of a screw jack for predictive maintenance

```
mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);

Serial.print("areal\t");

Serial.print(aaReal.x);

Serial.print("\t");

Serial.print(aaReal.y);

Serial.print("\t");

Serial.println(aaReal.z);

#endif

#ifdef OUTPUT_READABLE_WORLDACCEL

// display initial world-frame acceleration, adjusted to remove gravity
// and rotated based on known orientation from quaternion

mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetAccel(&aa, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);

Serial.print("aworld\t");

Serial.print(aaWorld.x);

Serial.print("\t");

Serial.print(aaWorld.y);

Serial.print("\t");

Serial.println(aaWorld.z);

#endif

#ifdef OUTPUT_TEAPOT

// display quaternion values in InvenSense Teapot demo format:

teapotPacket[2] = fifoBuffer[0];
teapotPacket[3] = fifoBuffer[1];
teapotPacket[4] = fifoBuffer[4];
teapotPacket[5] = fifoBuffer[5];
```

```

teapotPacket[6] = fifoBuffer[8];

teapotPacket[7] = fifoBuffer[9];

teapotPacket[8] = fifoBuffer[12];

teapotPacket[9] = fifoBuffer[13];

Serial.write(teapotPacket, 14);

teapotPacket[11]++; // packetCount, loops at 0xFF on purpose

#endif

// blink LED to indicate activity

blinkState = !blinkState;

digitalWrite(LED_PIN, blinkState);

}

}
    
```

- In order to integrate this code into Arduino UNO Board, this code has to be compiled and executed using the arrow symbol shown in the Fig 3.7

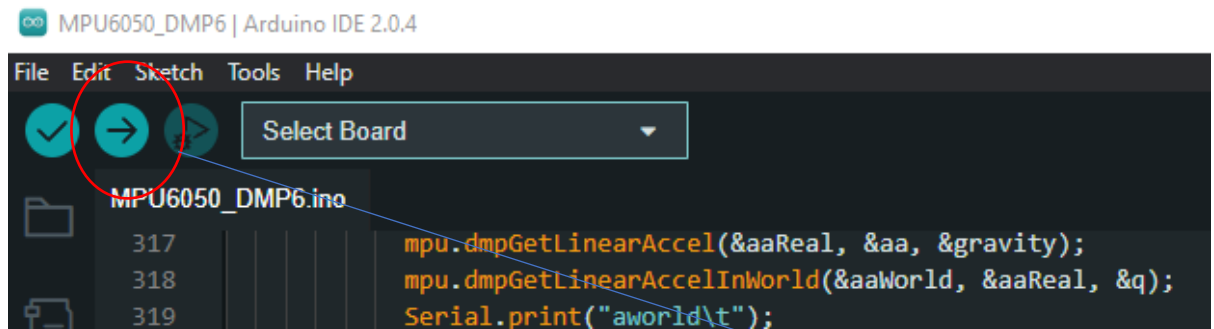
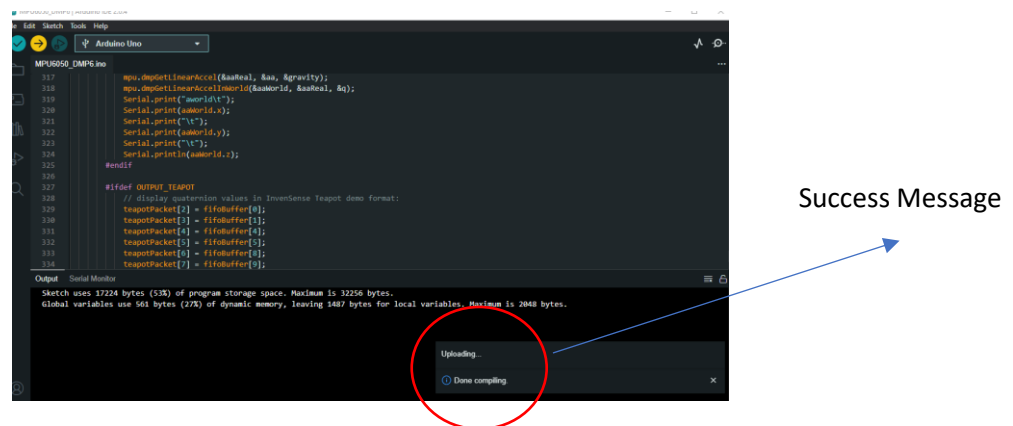


Fig 3.7(a) Compile button interface

Compile button

- Wait until the compilation is done and receive A success message in the terminal as shown in Fig 3.7(b)

Fig 3.7(b) Pop up message (Compiling done)



Processing Application



Fig 3.8 Processing application logo

- Download Java's Processing application from chrome and install it in the computer.
- Open the processing app and need to develop a code to integrate Arduino with processing application.
- The processing application's user interface will resemble that in Fig. 3.9

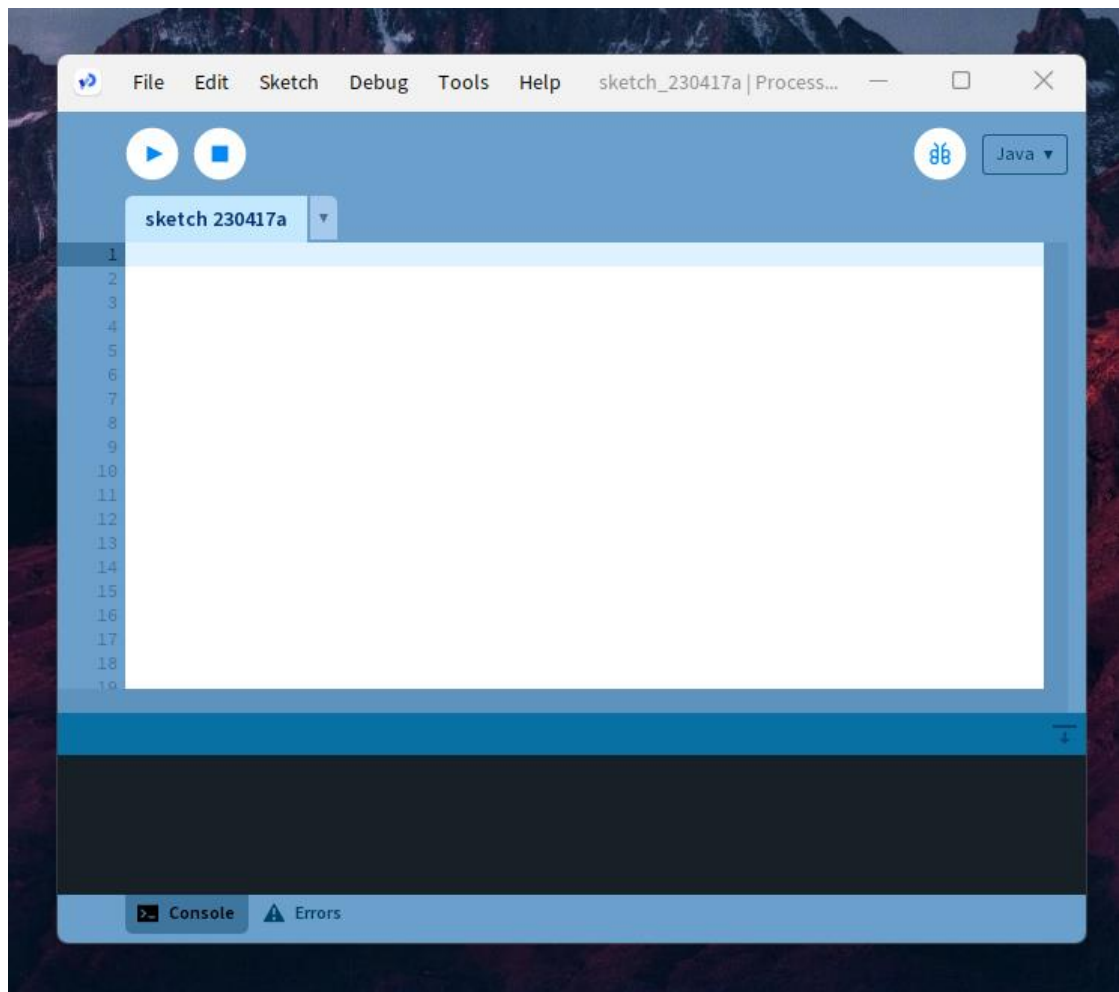


Fig 3.9 Interface of processing application

- Now a new extension called ‘TeaPot’ is to be downloaded which is a high-level JavaScript library for making SVG plots.
- Path to Open the TeaPot file from Libraries is :

C:\Users\CHANDRA MOULI\Documents\Processing\MPUTeapot

- The code developed is given below.

```
import processing.serial.*;
import processing.opengl.*;
import toxi.geom.*;
import toxi.processing.*;

// NOTE: requires ToxicLibs to be installed in order to run properly.
// 1. Download from http://toxiclibs.org/downloads
// 2. Extract into [userdir]/Processing/libraries
// (location may be different on Mac/Linux)
// 3. Run and bask in awesomeness
```



Building a digital twin of a screw jack for predictive maintenance

```
ToxiclibsSupport gfx;

Serial port;          // The serial port
char[] teapotPacket = new char[14]; // InvenSense Teapot packet
int serialCount = 0;   // current packet byte position
int synced = 0;
int interval = 0;

float[] q = new float[4];
Quaternion quat = new Quaternion(1, 0, 0, 0);

float[] gravity = new float[3];
float[] euler = new float[3];
float[] ypr = new float[3];

void setup() {
    // 300px square viewport using OpenGL rendering
    size(300, 300, OPENGL);
    gfx = new ToxiclibsSupport(this);

    // setup lights and antialiasing
    lights();
    smooth();

    // display serial port list for debugging/clarity
    println(Serial.list());

    // get the first available port (use EITHER this OR the specific port code below)
    //String portName = Serial.list()[0];

    // get a specific serial port (use EITHER this OR the first-available code above)
    String portName = "COM10";

    // open the serial port
    port = new Serial(this, portName, 115200);

    // send single character to trigger DMP init/start
    // (expected by MPU6050_DMP6 example Arduino sketch)
    port.write('r');
}

void draw() {
    if (millis() - interval > 1000) {
        // resend single character to trigger DMP init/start
        // in case the MPU is halted/reset while applet is running
        port.write('r');
        interval = millis();
    }

    // black background
    background(0);

    // translate everything to the middle of the viewport
    pushMatrix();
    translate(width / 2, height / 2);

    // 3-step rotation from yaw/pitch/roll angles (gimbal lock!)
    // ...and other weirdness I haven't figured out yet
    //rotateY(-ypr[0]);
    //rotateZ(-ypr[1]);
    //rotateX(-ypr[2]);

    // toxiclibs direct angle/axis rotation from quaternion (NO gimbal lock!)
    // (axis order [1, 3, 2] and inversion [-1, +1, +1] is a consequence of
    // different coordinate system orientation assumptions between Processing
    // and InvenSense DMP)
    float[] axis = quat.toAxisAngle();
    rotate(axis[0], -axis[1], axis[3], axis[2]);

    // draw main body in red
```



Building a digital twin of a screw jack for predictive maintenance

```
fill(255, 0, 0, 200);
box(10, 10, 200);

// draw front-facing tip in blue
fill(0, 0, 255, 200);
pushMatrix();
translate(0, 0, -120);
rotateX(PI/2);
drawCylinder(0, 20, 20, 8);
popMatrix();

// draw wings and tail fin in green
fill(0, 255, 0, 200);
beginShape(TRIANGLES);
vertex(-100, 2, 30); vertex(0, 2, -80); vertex(100, 2, 30); // wing top layer
vertex(-100, -2, 30); vertex(0, -2, -80); vertex(100, -2, 30); // wing bottom layer
vertex(-2, 0, 98); vertex(-2, -30, 98); vertex(-2, 0, 70); // tail left layer
vertex(2, 0, 98); vertex(2, -30, 98); vertex(2, 0, 70); // tail right layer
endShape();
beginShape(QUADS);
vertex(-100, 2, 30); vertex(-100, -2, 30); vertex(0, 2, -80); vertex(0, 2, -80);
vertex(100, 2, 30); vertex(100, -2, 30); vertex(0, -2, -80); vertex(0, -2, -80);
vertex(-100, 2, 30); vertex(-100, -2, 30); vertex(100, -2, 30); vertex(100, 2, 30);
vertex(-2, 0, 98); vertex(2, 0, 98); vertex(2, -30, 98); vertex(-2, -30, 98);
vertex(-2, 0, 70); vertex(2, 0, 70); vertex(2, 0, 70); vertex(-2, 0, 70);
vertex(-2, -30, 98); vertex(2, -30, 98); vertex(2, 0, 70); vertex(-2, 0, 70);
endShape();

popMatrix();
}

void serialEvent(Serial port) {
  interval = millis();
  while (port.available() > 0) {
    int ch = port.read();

    if (syncd == 0 && ch != '$') return; // initial synchronization - also used to resync/realign if needed
    syncd = 1;
    print ((char)ch);

    if ((serialCount == 1 && ch != 2)
        || (serialCount == 12 && ch != '\r')
        || (serialCount == 13 && ch != '\n')) {
      serialCount = 0;
      syncd = 0;
      return;
    }

    if (serialCount > 0 || ch == '$') {
      teapotPacket[serialCount++] = (char)ch;
      if (serialCount == 14) {
        serialCount = 0; // restart packet byte position

        // get quaternion from data packet
        q[0] = ((teapotPacket[2] << 8) | teapotPacket[3]) / 16384.0f;
        q[1] = ((teapotPacket[4] << 8) | teapotPacket[5]) / 16384.0f;
        q[2] = ((teapotPacket[6] << 8) | teapotPacket[7]) / 16384.0f;
        q[3] = ((teapotPacket[8] << 8) | teapotPacket[9]) / 16384.0f;
        for (int i = 0; i < 4; i++) if (q[i] >= 2) q[i] = -4 + q[i];

        // set our totilibs quaternion to new data
        quat.set(q[0], q[1], q[2], q[3]);

        /*
        // below calculations unnecessary for orientation only using totilibs

        // calculate gravity vector
        gravity[0] = 2 * (q[1]*q[3] - q[0]*q[2]);
        gravity[1] = 2 * (q[0]*q[1] + q[2]*q[3]);
        gravity[2] = q[0]*q[0] - q[1]*q[1] - q[2]*q[2] + q[3]*q[3];
        */
      }
    }
  }
}
```



Building a digital twin of a screw jack for predictive maintenance

```
// calculate Euler angles
euler[0] = atan2(2*q[1]*q[2] - 2*q[0]*q[3], 2*q[0]*q[0] + 2*q[1]*q[1] - 1);
euler[1] = -asin(2*q[1]*q[3] + 2*q[0]*q[2]);
euler[2] = atan2(2*q[2]*q[3] - 2*q[0]*q[1], 2*q[0]*q[0] + 2*q[3]*q[3] - 1);

// calculate yaw/pitch/roll angles
ypr[0] = atan2(2*q[1]*q[2] - 2*q[0]*q[3], 2*q[0]*q[0] + 2*q[1]*q[1] - 1);
ypr[1] = atan(gravity[0] / sqrt(gravity[1]*gravity[1] + gravity[2]*gravity[2]));
ypr[2] = atan(gravity[1] / sqrt(gravity[0]*gravity[0] + gravity[2]*gravity[2]));

// output various components for debugging
//println("q:\t" + round(q[0]*100.0f)/100.0f + "\t" + round(q[1]*100.0f)/100.0f + "\t" + round(q[2]*100.0f)/100.0f +
"\t" + round(q[3]*100.0f)/100.0f);
//println("euler:\t" + euler[0]*180.0f/PI + "\t" + euler[1]*180.0f/PI + "\t" + euler[2]*180.0f/PI);
//println("ypr:\t" + ypr[0]*180.0f/PI + "\t" + ypr[1]*180.0f/PI + "\t" + ypr[2]*180.0f/PI);
*/
}
}
}

void drawCylinder(float topRadius, float bottomRadius, float tall, int sides) {
    float angle = 0;
    float angleIncrement = TWO_PI / sides;
    beginShape(QUAD_STRIP);
    for (int i = 0; i < sides + 1; ++i) {
        vertex(topRadius*cos(angle), 0, topRadius*sin(angle));
        vertex(bottomRadius*cos(angle), tall, bottomRadius*sin(angle));
        angle += angleIncrement;
    }
    endShape();

    // If it is not a cone, draw the circular top cap
    if (topRadius != 0) {
        angle = 0;
        beginShape(TRIANGLE_FAN);

        // Center point
        vertex(0, 0, 0);
        for (int i = 0; i < sides + 1; i++) {
            vertex(topRadius * cos(angle), 0, topRadius * sin(angle));
            angle += angleIncrement;
        }
        endShape();
    }

    // If it is not a cone, draw the circular bottom cap
    if (bottomRadius != 0) {
        angle = 0;
        beginShape(TRIANGLE_FAN);

        // Center point
        vertex(0, tall, 0);
        for (int i = 0; i < sides + 1; i++) {
            vertex(bottomRadius * cos(angle), tall, bottomRadius * sin(angle));
            angle += angleIncrement;
        }
        endShape();
    }
}
```

- Hence all the required codes are developed and execution part is left.
- Execute the file using execute button in processing application.

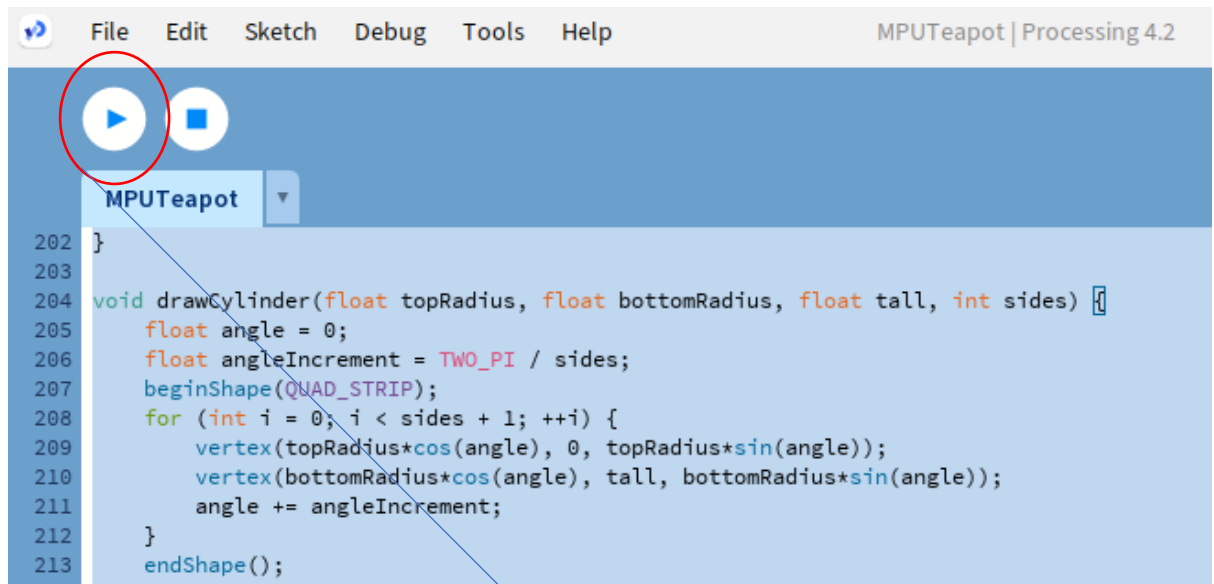


Fig 3.10 Execute button interface

Execute Button

- When execute button is clicked a blank screen with the 3d model will be opened which is running live as shown in the Fig 3.11

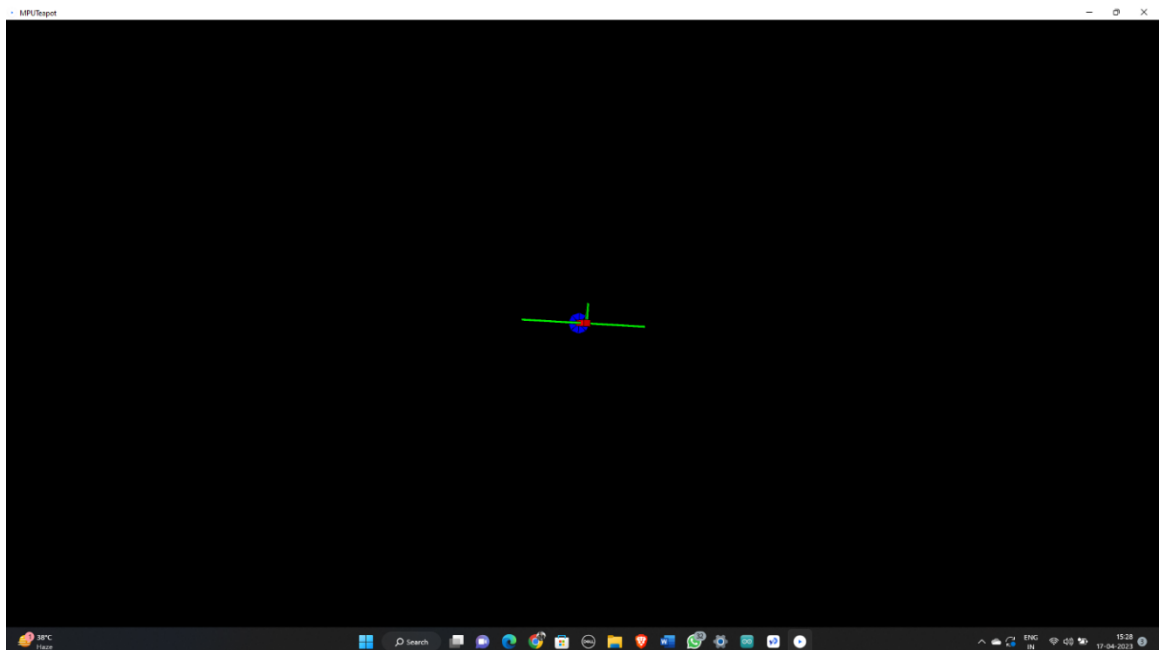


Fig 3.11 Space containing 3d model

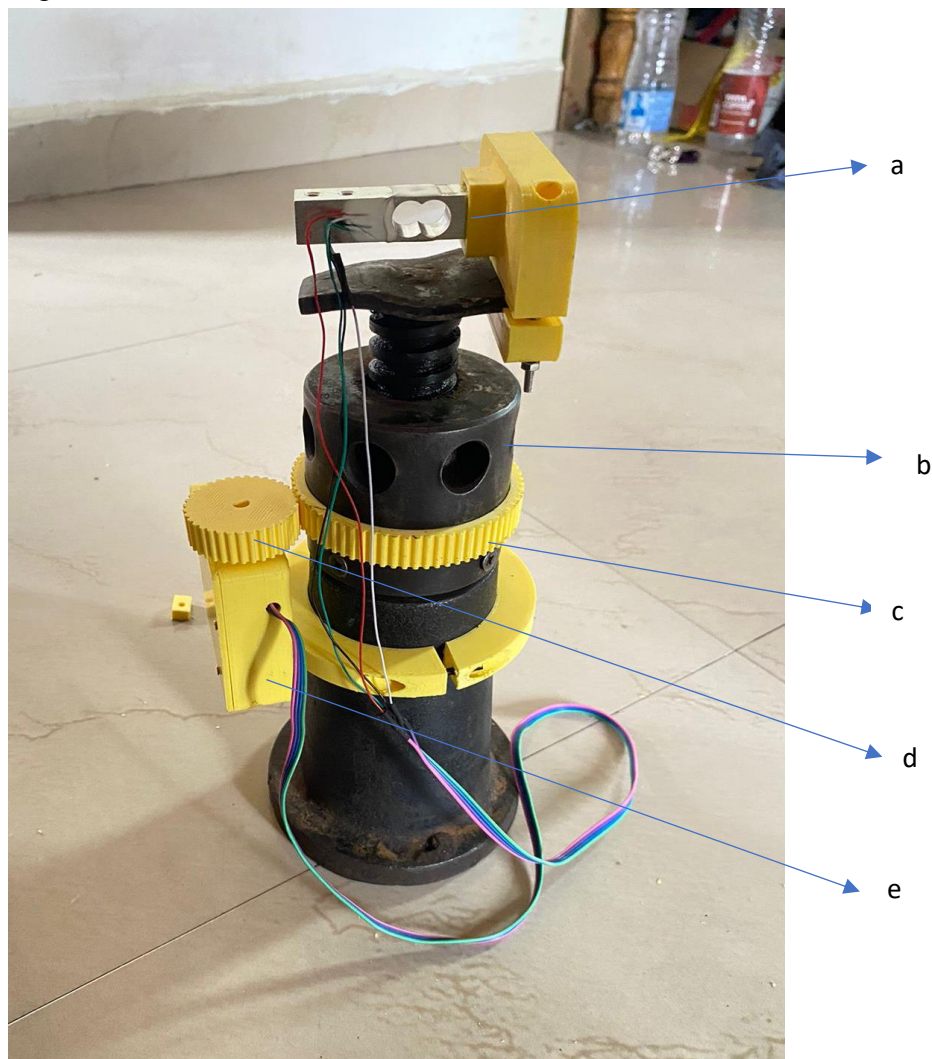
- Hence the 3d model is integrated with the sensor equipped to the physical object and will be resembled as a twin to the physical object.

3.2 Development of Digital Twin of a screw jack using three.js, Fusion 360 and webapp.

3.2.1. Experimental setup

Experimental setup of a physical screw jack is to be done as shown in the figure given below

Fig 3.12



- a. 5kg Load cell
- b. Screw jack
- c. Spur Gear attached to screw jack
- d. Spur gear attached to the magnetic encoder
- e. Sensor box



3.2.1.1 Construction:

- Screw jack has to be taken and make sure that it is in proper working condition.
- Around the head of the screw jack, a spur gear consisting of 70 teeth is attached.
- A load cell of 5kg capacity is attached to the carriage of the screw jack to read load input.
- Using 3d printed components sensor box is attached, which is equipped with magnetic encoder and microcontroller inside it.
- Another spur gear consisting of 35 teeth is kept above the sensor box by fixing a magnetic coil to it.
- Connect all the components using cables and make sure the connections are right.

3.2.1.2 Building a sensor box:

- Sensor box consists of 2 components:
 1. Microcontroller
 2. Magnetic encoder
- Entire rectangular box should be 3d printed in order to equip components into it.
- Microcontroller is connected with load cell and magnetic encoder just like as shown in the pinout diagrams shown below.

3.2.2. Designing and modelling

- Using Autodesk Fusion 360 (For personal use), two 3-D models are created.
- One model is of Screw Jack and another is of sensor box consisting of sensors, microcontroller and magnetic encoder.
- Models created are in 3df format.
- Now using URDF exporter, created 3df models needed to be exported into URDF format.



3.2.2.1. URDF Format of screw jack

```

<robot name="jack" >

<link name="base_link">
  <inertial>
    <origin xyz="0.0030071865378406538 0.00014154788942299552 0.05591757888982463" rpy="0
0 0"/>
    <mass value="4.632745962530948"/>
    <inertia ixx="0.009676" iyy="0.009676" izz="0.005056" ixy="0.0" iyz="0.0" ixz="-0.0"/>
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <mesh filename="./meshes/base_link.stl" scale="0.001 0.001 0.001"/>
    </geometry>
    <material name="silver">
      <color rgba="0.700 0.700 0.700 1.000"/>
    </material>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <mesh filename="./meshes/base_link.stl" scale="0.001 0.001 0.001"/>
    </geometry>
  </collision>
</link>

<link name="head_1">
  <inertial>
    <origin xyz="-1.77956820907434e-06 1.1441140199030988e-06 0.03063938447984857" rpy="0 0
0"/>
    <mass value="1.5495145241048947"/>
    <inertia ixx="0.001131" iyy="0.001131" izz="0.001195" ixy="-0.0" iyz="-0.0" ixz="-0.0"/>
  </inertial>
  <visual>
    <origin xyz="-0.003007 -0.000142 -0.132021" rpy="0 0 0"/>
    <geometry>
      <mesh filename="./meshes/head_1.stl" scale="0.001 0.001 0.001"/>
    </geometry>
    <material name="silver">
      <color rgba="0.700 0.700 0.700 1.000"/>
    </material>
  </visual>
  <collision>
    <origin xyz="-0.003007 -0.000142 -0.132021" rpy="0 0 0"/>
    <geometry>
      <mesh filename="./meshes/head_1.stl" scale="0.001 0.001 0.001"/>
    </geometry>
  </collision>
</link>

<link name="screw_1">
  <inertial>
    <origin xyz="-0.0031033581934499896 0.196039434773979 0.14856469037947262" rpy="0 0 0"/>
    <mass value="0.7567949663178379"/>
    <inertia ixx="0.002474" iyy="0.002516" izz="0.000156" ixy="0.0" iyz="-1e-06" ixz="5e-06"/>
  </inertial>

```




Building a digital twin of a screw jack for predictive maintenance

```
<visual>
  <origin xyz="-0.006014 0.19588 -0.001163" rpy="0 0 0"/>
  <geometry>
    <mesh filename="./meshes/screw_1.stl" scale="0.001 0.001 0.001"/>
  </geometry>
  <material name="silver">
    <color rgba="0.700 0.700 0.700 1.000"/>
  </material>
</visual>
<collision>
  <origin xyz="-0.006014 0.19588 -0.001163" rpy="0 0 0"/>
  <geometry>
    <mesh filename="./meshes/screw_1.stl" scale="0.001 0.001 0.001"/>
  </geometry>
</collision>
</link>

<link name="bull_gear_1">
  <inertial>
    <origin xyz="-0.0030068134621596887 0.13402154788942292 0.15150819039459762" rpy="0 0 0"/>
    <mass value="0.014071732363726169"/>
    <inertia ixx="1.1e-05" iyy="1.1e-05" izz="2.2e-05" ixy="0.0" iyz="0.0" ixz="0.0"/>
  </inertial>
  <visual>
    <origin xyz="-0.006014 0.13388 -0.001163" rpy="0 0 0"/>
    <geometry>
      <mesh filename="./meshes/bull_gear_1.stl" scale="0.001 0.001 0.001"/>
    </geometry>
    <material name="black">
      <color rgba="0.1 0.1 0.1 1.000"/>
    </material>
  </visual>
  <collision>
    <origin xyz="-0.006014 0.13388 -0.001163" rpy="0 0 0"/>
    <geometry>
      <mesh filename="./meshes/bull_gear_1.stl" scale="0.001 0.001 0.001"/>
    </geometry>
  </collision>
</link>

<link name="pinion_gear_1">
  <inertial>
    <origin xyz="-0.058654699221435534 -0.0005762682736991168 -0.13916623482795912" rpy="0 0 0"/>
    <mass value="0.015611687834022035"/>
    <inertia ixx="2e-06" iyy="2e-06" izz="3e-06" ixy="0.0" iyz="0.0" ixz="0.0"/>
  </inertial>
  <visual>
    <origin xyz="-0.059619 -0.061736 -0.293892" rpy="0 0 0"/>
    <geometry>
      <mesh filename="./meshes/pinion_gear_1.stl" scale="0.001 0.001 0.001"/>
    </geometry>
    <material name="black">
      <color rgba="0.1 0.1 0.1 1.000"/>
    </material>
  </visual>
  <collision>
    <origin xyz="-0.059619 -0.061736 -0.293892" rpy="0 0 0"/>
    <geometry>
```



```

    <mesh filename="/meshes/pinion_gear_1.stl" scale="0.001 0.001 0.001"/>
  </geometry>
</collision>
</link>

<link name="sensor_1">
  <inertial>
    <origin xyz="0.0036287929269774154 0.1692092556548879 0.10603625338614514" rpy="0 0
0"/>
    <mass value="0.6833126827436713"/>
    <inertia ixx="0.00126" iyy="0.000709" izz="0.001565" ixy="-0.000147" iyz="-6e-05" ixz="-1.3e-
05"/>
  </inertial>
  <visual>
    <origin xyz="-0.006014 0.13388 -0.001163" rpy="0 0 0"/>
    <geometry>
      <mesh filename="/meshes/sensor_1.stl" scale="0.001 0.001 0.001"/>
    </geometry>
    <material name="yellow">
      <color rgba="1.0 0.700 0.0 1.000"/>
    </material>
  </visual>
  <collision>
    <origin xyz="-0.006014 0.13388 -0.001163" rpy="0 0 0"/>
    <geometry>
      <mesh filename="/meshes/sensor_1.stl" scale="0.001 0.001 0.001"/>
    </geometry>
  </collision>
</link>

<link name="load_cell_holder_1">
  <inertial>
    <origin xyz="-0.00809316093830615 -3.896582473705639e-08 0.005959800511437602" rpy="0 0
0"/>
    <mass value="0.060378076940754415"/>
    <inertia ixx="4.7e-05" iyy="1.6e-05" izz="3.4e-05" ixy="-0.0" iyz="-0.0" ixz="0.0"/>
  </inertial>
  <visual>
    <origin xyz="-0.040039 -0.000234 -0.222021" rpy="0 0 0"/>
    <geometry>
      <mesh filename="/meshes/load_cell_holder_1.stl" scale="0.001 0.001 0.001"/>
    </geometry>
    <material name="black">
      <color rgba="0.1 0.1 0.1 1.000"/>
    </material>
  </visual>
  <collision>
    <origin xyz="-0.040039 -0.000234 -0.222021" rpy="0 0 0"/>
    <geometry>
      <mesh filename="/meshes/load_cell_holder_1.stl" scale="0.001 0.001 0.001"/>
    </geometry>
  </collision>
</link>

<link name="load_cell_1">
  <inertial>
    <origin xyz="-0.03787910394686526 0.01611896104433127 -0.010349809605404992" rpy="0 0
0"/>
    <mass value="0.029241655696833388"/>
    <inertia ixx="1e-06" iyy="1.8e-05" izz="1.8e-05" ixy="0.0" iyz="0.0" ixz="-0.0"/>

```



Building a digital twin of a screw jack for predictive maintenance

```
</inertial>
<visual>
  <origin xyz="-0.037451 0.015885 -0.248871" rpy="0 0 0"/>
  <geometry>
    <mesh filename="/meshes/load_cell_1.stl" scale="0.001 0.001 0.001"/>
  </geometry>
  <material name="silver">
    <color rgba="0.700 0.700 0.700 1.000"/>
  </material>
</visual>
<collision>
  <origin xyz="-0.037451 0.015885 -0.248871" rpy="0 0 0"/>
  <geometry>
    <mesh filename="/meshes/load_cell_1.stl" scale="0.001 0.001 0.001"/>
  </geometry>
</collision>
</link>

<joint name="head" type="revolute">
  <origin xyz="0.003007 0.000142 0.132021" rpy="0 0 0"/>
  <parent link="base_link"/>
  <child link="head_1"/>
  <axis xyz="0.0 0.0 -1.0"/>
  <limit upper="100000" lower="" effort="100" velocity="100"/>
</joint>

<joint name="screw" type="prismatic">
  <origin xyz="0.003007 -0.196022 00.0" rpy="0 0 0"/>
  <parent link="base_link"/>
  <child link="screw_1"/>
  <axis xyz="-0.0 -0.0 1.0"/>
  <limit upper="0.12" lower="0.0" effort="100" velocity="100"/>
</joint>

<joint name="Rigid 3" type="fixed">
  <origin xyz="0.003007 -0.134022 -0.130858" rpy="0 0 0"/>
  <parent link="head_1"/>
  <child link="bull_gear_1"/>
</joint>

<joint name="pinion_gear" type="continuous">
  <origin xyz="0.053605 0.195616 0.291" rpy="0 0 0"/>
  <parent link="sensor_1"/>
  <child link="pinion_gear_1"/>
  <axis xyz="-0.0 0.0 1.0"/>
</joint>

<joint name="Rigid 12" type="fixed">
  <origin xyz="0.003007 -0.134022 0.001" rpy="0 0 0"/>
  <parent link="base_link"/>
  <child link="sensor_1"/>
</joint>

<joint name="Rigid 13" type="fixed">
  <origin xyz="0.034025 0.196114 0.220858" rpy="0 0 0"/>
  <parent link="screw_1"/>
  <child link="load_cell_holder_1"/>
</joint>

<joint name="Rigid 14" type="fixed">
```



```
<origin xyz="-0.002588 -0.016119 0.02685" rpy="0 0 0"/>
<parent link="load_cell_holder_1"/>
<child link="load_cell_1"/>
</joint>

</robot>
```

3.2.3. Executing The Web Application

- Using Visual studio code, backend architecture for inputting serial communication is developed in javascript.
- Entire architecture is developed into two components
 1. Frontend
 2. Backend
- Backend consists of Source code that integrates 3D Model and The physical screw jack.
- Frontend consists of 3D Model that replicates the physical screw jack, simply the digital twin of screw jack.
- Frontend is also known as client and Backend is also known as Server.
- Entire backend is developed using java script

Backend Code

Index.js

```
const { SerialPort } = require("serialport");
const { DelimiterParser } = require("@serialport/parser-delimiter");

const express = require("express");
const app = express();
const http = require("http");
const server = http.createServer(app);
const { Server } = require("socket.io")

var comport = undefined;
var port = undefined;
```



Building a digital twin of a screw jack for predictive maintenance

```
var parser = undefined;

SerialPort.list().then((devices) => {
  devices.forEach((device) => {
    // vendorId: '10C4',
    // productId: 'EA60'
    if (device.vendorId == "10C4" && device.productId == "EA60") {
      comport = device.path;
    }
  });
  console.log(comport);
  if (comport != undefined) {
    port = new SerialPort({ path: comport, baudRate: 115200 });
    parser = port.pipe(new DelimiterParser({ delimiter: "\n" }));
    const io = new Server(server);
    io.on("connection", (socket) => {
      console.log("a user connected");
    });
    server.listen(3000, () => {
      console.log("listening on *:3000");
    });
    parser.on("data", (line) => {
      console.log(`Received data from serial port: ${line}`);
      io.emit("jack", line );
    });
    process.stdin.resume();
    // Keep the script running until a SIGINT signal is received (i.e. the user presses ctrl+c)
    process.on("SIGINT", () => {
      console.log("Exiting...");
      process.exit();
    });
  }
});
```



Package.json

```
{
  "name": "digital_twin_backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "serialport": "^10.5.0",
    "socket.io": "^4.6.1"
  }
}
```

Frontend Code

```
<template>
  <div ref="container">
    <div id="data">
      <h2>Height:</h2>
      <p>{{ height }} <span>mm</span></p>
      <h2>Weight:</h2>
      <p :class="{red:weight>1000}">{{ weight }} <span>g</span></p>
    </div>
  </div>
</template>

<script>
import URDFLoader from "urdf-loader";
import {
  WebGLRenderer,
  PerspectiveCamera,
```



Building a digital twin of a screw jack for predictive maintenance

```
Scene,
Mesh,
PlaneBufferGeometry,
ShadowMaterial,
DirectionalLight,
PCFSoftShadowMap,
sRGBEncoding,
Color,
AmbientLight,
// Box3,
// Vector3,
LoadingManager,
// MeshPhongMaterial,
} from "three";
import { OrbitControls } from "three/examples/jsm/controls/OrbitControls.js";
import io from "socket.io-client";
let thescene;
export default {
  name: "App",

  data() {
    return {
      scene: null,
      camera: null,
      renderer: null,
      robot: null,
      controls: null,
      socket: io("http://localhost:3000", { transports: ["websocket"] }),
      height: 0.0,
      angle: 0.0,
      weight: 0.0,
      max_weight: 1000,
    };
  },

  components: { },
```



Building a digital twin of a screw jack for predictive maintenance

```
mounted() {
  this.init();
  this.animate();
  let vm = this;
  this.socket.on("jack", (data) => {
    var buffer = new Uint8Array(data);
    var str = String.fromCharCode.apply(null, buffer);
    const values = str.split(";");
    vm.angle = Number(values[0]);
    vm.height = Number(values[1]);
    vm.weight = Number(values[2]) < 0 ? 0 : Number(values[2]);
    vm.robot.setJointValue("screw", vm.height / 1500);
    vm.robot.setJointValue("head", vm.angle/100);
    var color_ratio = vm.weight/vm.max_weight;
    if(color_ratio>1.0)color_ratio = 1.0;
    var hue =(1 - color_ratio) * 120
    var color = new Color(`hsl(${hue}, 100%, 50%)`);
    thescene.traverse(function (child) {
      if (child.isMesh) {
        if (child.parent.isURDFVisual) {
          if (child.parent.parent.name == "load_cell_1") {
            child.material.color.setHex("0x"+color.getHexString());
          }
        }
      }
    });
  });
},
methods: {
  animate() {
    let vm = this;
    requestAnimationFrame(vm.animate);
    vm.controls.update();
    vm.renderer.render(thescene, vm.camera);
  },
  init() {
```




Building a digital twin of a screw jack for predictive maintenance

```
let vm = this;
thescene = new Scene();
thescene.background = new Color(0x263238);

vm.camera = new PerspectiveCamera(3, window.devicePixelRatio, 0.1, 10000);
vm.camera.position.set(6, 5, -20);
vm.camera.translateZ(12);
vm.camera.lookAt(0, 0, 0);

vm.renderer = new WebGLRenderer({
  antialias: false,
});
vm.renderer.setPixelRatio(window.devicePixelRatio);
vm.renderer.setSize(window.innerWidth, window.innerHeight);
vm.renderer.outputEncoding = sRGBEncoding;
vm.renderer.shadowMap.enabled = true;
vm.renderer.shadowMap.type = PCFSoftShadowMap;
this.$refs.container.appendChild(vm.renderer.domElement);

const directionalLight = new DirectionalLight(0xffffff, 1.0);
directionalLight.castShadow = true;
directionalLight.shadow.mapSize.setScalar(1024);
directionalLight.position.set(5, 30, 5);
thescene.add(directionalLight);

const ambientLight = new AmbientLight(0xffffff, 0.2);
thescene.add(ambientLight);

const ground = new Mesh(
  new PlaneBufferGeometry(),
  new ShadowMaterial({ opacity: 0.25 })
);
ground.rotation.x = -Math.PI / 2;
ground.scale.setScalar(30);
ground.receiveShadow = true;
thescene.add(ground);
```



Building a digital twin of a screw jack for predictive maintenance

```
vm.controls = new OrbitControls(vm.camera, vm.renderer.domElement);
vm.controls.minDistance = 1;
vm.controls.target.y = 0;
vm.controls.update();
vm.renderer.render(thescene, vm.camera);

const manager = new LoadingManager();
const loader = new URDFLoader(manager);

loader.load(
  "./jack_description/jack.URDF", // The path to the URDF within the package OR absolute
  (robot) => {
    robot.rotation.x = -Math.PI / 2;
    robot.rotation.z = -1;
    vm.robot = robot;
    thescene.add(robot);
  }
);

manager.onLoad = () => {
  let vm = this;
  vm.robot.traverse((c) => {
    c.castShadow = true;
  });

  vm.robot.updateMatrixWorld(true);
  vm.scene = thescene;
  vm.renderer.render(thescene, vm.camera);
  // thescene.traverse(function (child) {
  //   if(child.isMesh){
  //     if(child.parent.isURDFVisual){
  //       if(child.parent.parent.name === "load_cell_1"){
  //         child.material.color.setHex(0xff0000);
  //       }
  //     }
  //   }
  // });
```



Building a digital twin of a screw jack for predictive maintenance

```
// });  
  
// setInterval(() => {  
  
    // vm.robot.setJointValue("screw", 0.06);  
  
    // }, 5000);  
};  
},  
},  
};  
</script>  
  
<style>  
#app {  
    font-family: Avenir, Helvetica, Arial, sans-serif;  
    -webkit-font-smoothing: antialiased;  
    -moz-osx-font-smoothing: grayscale;  
    text-align: center;  
    color: #2c3e50;  
    margin-top: 60px;  
}  
canvas {  
    position: absolute;  
    top: 10vh;  
    width: 100vh !important;  
    height: 80vh !important;  
}  
body {  
    background-color: #263238;  
}  
#data {  
    position: absolute;  
    z-index: 99;  
    top: 30%;  
    right: 20%;
```



Building a digital twin of a screw jack for predictive maintenance

```
    min-width: 17vw;
  }
h2 {
  margin: 0 0 5px 0;
  font-size: 40px;
  color: white;
  opacity: 0.7;
}
p {
  margin: 0 0 15px 0;
  font-size: 50px;
  font-weight: bold;
  color: RGB(255, 180, 0);
}
p > span {
  margin: 0 0 0 5px;
  font-size: 30px;
  color: white;
  opacity: 0.7;
}
.red{
  color: RGB(255, 0, 0);
}
.yellow{
  color: RGB(255, 180, 0);
}
</style>
```

Background.js

```
'use strict'

import { app, protocol, BrowserWindow } from 'electron'
import { createProtocol } from 'vue-cli-plugin-electron-builder/lib'
import installExtension, { VUEJS_DEVTOOLS } from 'electron-devtools-installer'
const isDevelopment = process.env.NODE_ENV !== 'production'

// Scheme must be registered before the app is ready
```



Building a digital twin of a screw jack for predictive maintenance

```
protocol.registerSchemesAsPrivileged([
  { scheme: 'app', privileges: { secure: true, standard: true } }
])

async function createWindow() {
  // Create the browser window.
  const win = new BrowserWindow({
    title: "Digital Twin",
    maximizable: false,
    resizable: false,
    webPreferences: {

      // Use pluginOptions.nodeIntegration, leave this alone
      // See nklayman.github.io/vue-cli-plugin-electron-builder/guide/security.html#node-integration for more
      info
      nodeIntegration: process.env.ELECTRON_NODE_INTEGRATION,
      contextIsolation: !process.env.ELECTRON_NODE_INTEGRATION
    }
  })
  win.setMenu(null)
  win.once('ready-to-show', () => {
    win.maximize()
  })

  if (process.env.WEBPACK_DEV_SERVER_URL) {
    // Load the url of the dev server if in development mode
    await win.loadURL(process.env.WEBPACK_DEV_SERVER_URL)
    if (!process.env.IS_TEST) win.webContents.openDevTools()
  } else {
    createProtocol('app')
    // Load the index.html when not in development
    win.loadURL('app://./index.html')
  }
}

// Quit when all windows are closed.
```



Building a digital twin of a screw jack for predictive maintenance

```
app.on('window-all-closed', () => {
  // On macOS it is common for applications and their menu bar
  // to stay active until the user quits explicitly with Cmd + Q
  if (process.platform !== 'darwin') {
    app.quit()
  }
})

app.on('activate', () => {
  // On macOS it's common to re-create a window in the app when the
  // dock icon is clicked and there are no other windows open.
  if (BrowserWindow.getAllWindows().length === 0) createWindow()
})

// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Some APIs can only be used after this event occurs.
app.on('ready', async () => {
  if (isDevelopment && !process.env.IS_TEST) {
    // Install Vue Devtools
    try {
      await installExtension(VUEJS_DEVTOOLS)
    } catch (e) {
      console.error('Vue Devtools failed to install:', e.toString())
    }
  }
  createWindow()
})

// Exit cleanly on request from parent process in development mode.
if (isDevelopment) {
  if (process.platform === 'win32') {
    process.on('message', (data) => {
      if (data === 'graceful-exit') {
        app.quit()
      }
    })
  }
}
```



Building a digital twin of a screw jack for predictive maintenance

```
  })
} else {
  process.on('SIGTERM', () => {
    app.quit()
  })
}
}
```

Main.js

```
import Vue from 'vue'
import App from './App.vue'
Vue.config.productionTip = false
new Vue({
  render: h => h(App),
}).$mount('#app')
```

Arduino Code

```
#include "HX711.h"
#include "AS5600.h"
#include "Wire.h"
AS5600 as5600; // use default Wire
long enc_offset = 0;
long weight = 0;
float enc_height_resolution = 0.00097656; //8/8192 =0.00195312;
float enc_ang_resolution = 0.04394531; //360/8192=0.04394531
// HX711 circuit wiring
const int LOADCELL_DOUT_PIN = 12;
const int LOADCELL_SCK_PIN = 13;
long lastTime = millis();
long last_send = millis();
long ticks = 0;
HX711 scale;
void setup() {
```



Building a digital twin of a screw jack for predictive maintenance

```
// Setup Serial Monitor
Serial.begin(115200);

scale.begin(LoadCELL_DOUT_PIN, LoadCELL_SCK_PIN);

scale.set_scale(454.126984);          // this value is obtained by calibrating the scale with known weights;
see the README for details

scale.tare();

as5600.begin(14); // set direction pin.

as5600.setDirection(AS5600_COUNTERCLOCK_WISE); // default, just be explicit.

delay(1000);

enc_offset = as5600.getCumulativePosition();
}

void loop() {
  if (millis() - lastTime >= 100)
  {
    lastTime = millis();
    ticks = as5600.getCumulativePosition() - enc_offset;
  }

  if (scale.is_ready()) {
    weight = scale.get_units();
    // Serial.print("HX711 reading: ");
    // Serial.println(weight);
  }

  if (millis() - last_send >= 200) {
    float height = ticks * enc_height_resolution;
    if (height < 0) height = 0;
    float ang = ticks * enc_ang_resolution;
    if (ang < 0) ang = 0;
    String output_str = String(ang) + ";" + String(height) + ";" + String(weight);
    Serial.println(output_str);
    last_send = millis();
  }

  delay(1);
}
```




3.2.4. Procedure to Execute

- All the source codes are saved in the PC or Laptop
- In order to execute firstly command line need to be opened.
- Two command line windows are opened, one is to run frontend and other one to run backend.
- In the backend command line, change the directory to backend using ~ cd command.
Command for backend is : `node index.js`
- In the frontend command line, change the directory to frontend using ~ cd command.
Command for Frontend is : `npm run electron:serve`



CHAPTER 4

RESULTS & DISCUSSIONS

4.1 Results of Digital twin model using Arduino and Processing application.

This is the final 3d model of a physical object in virtual/ Digital format. Which exactly possess the structural and functional features of physical object.

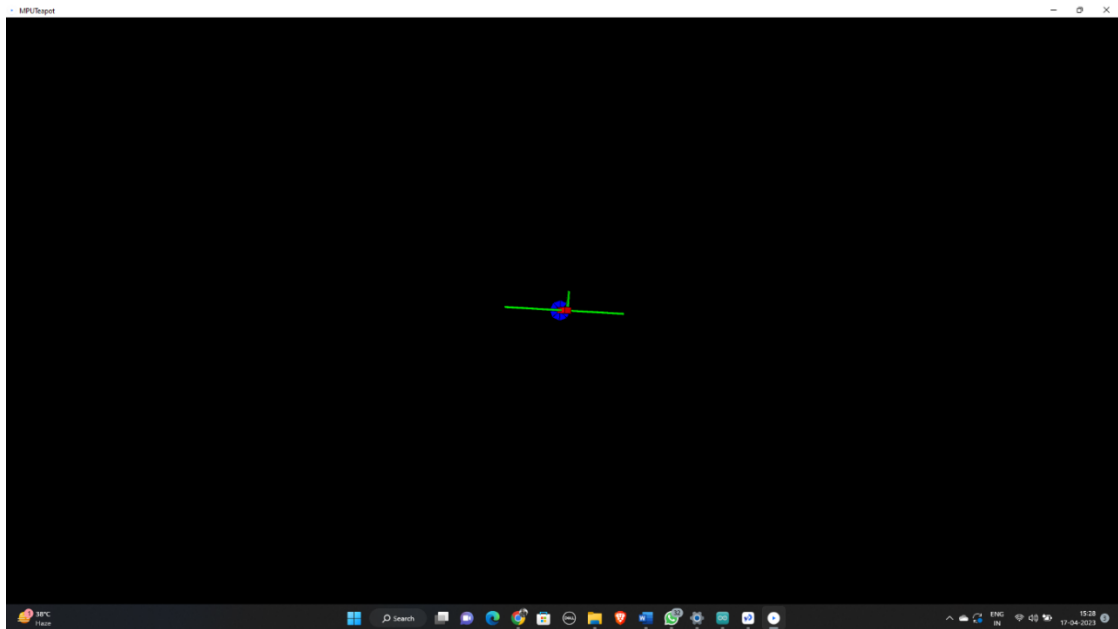


Fig 4.1 : Digital twin of physical object in virtual space.

Following figures Fig 4.2, Fig 4.3, Fig 4.4 and Fig 4.5 Depict the variation or transformation of 3d model based on the movement of physical object.

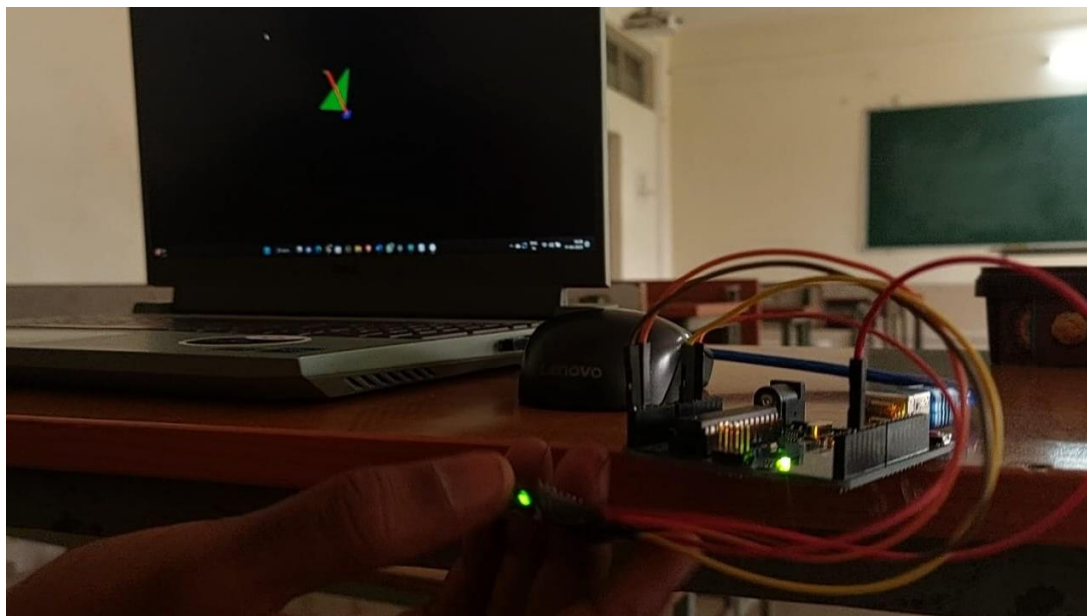


Fig 4.2 Left orientation of 3d object

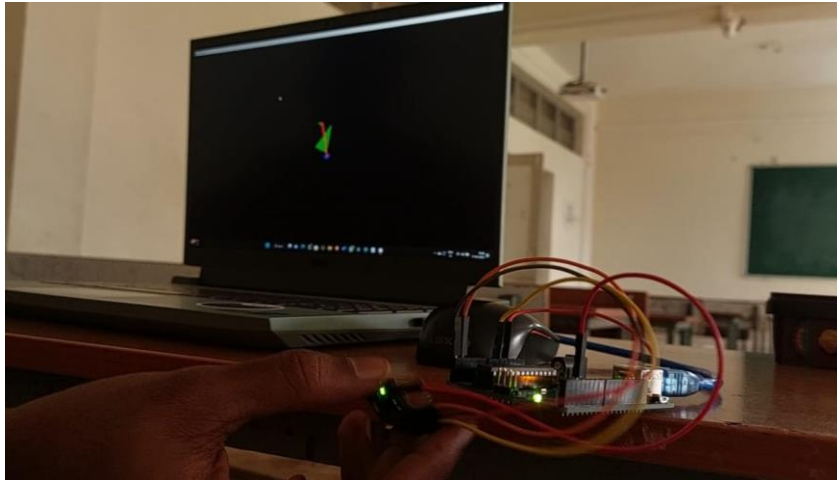


Fig 4.3 Right orientation of 3d object

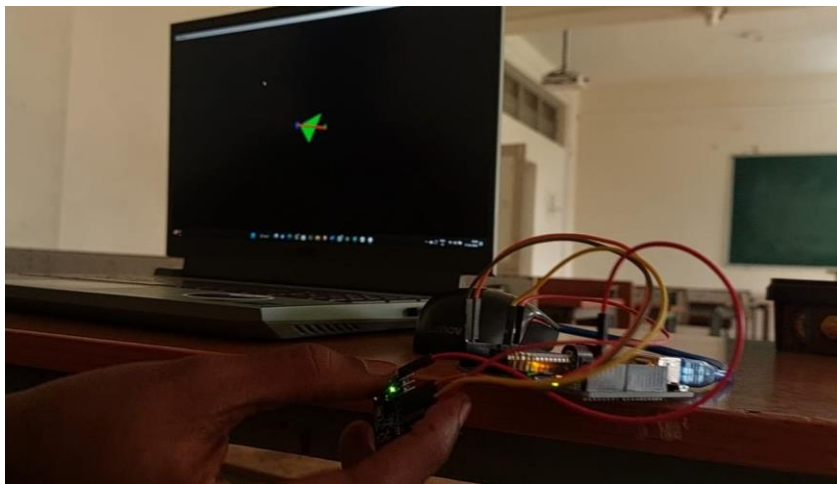


Fig 4.4 Top view of 3d object



Fig 4.5 Bottom view of 3d object

4.2 Results of Digital Twin of a screw jack developed in fusion 360 and implemented in webapp using three.js



Fig 4.6 Front view of Screw jack developed in fusion 360



Fig 4.7 3d view of Screw jack developed in fusion 360

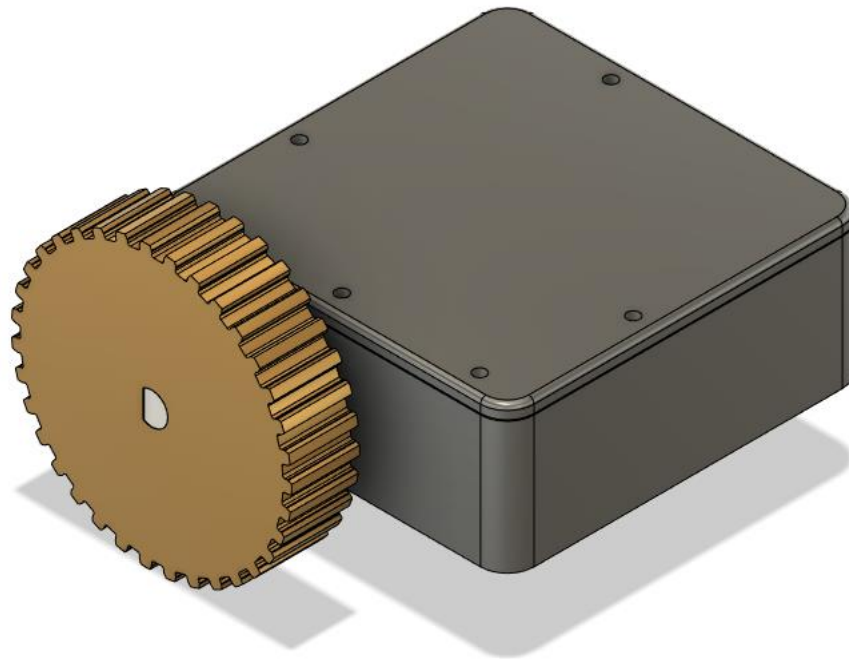


Fig 4.8 Sensor Box

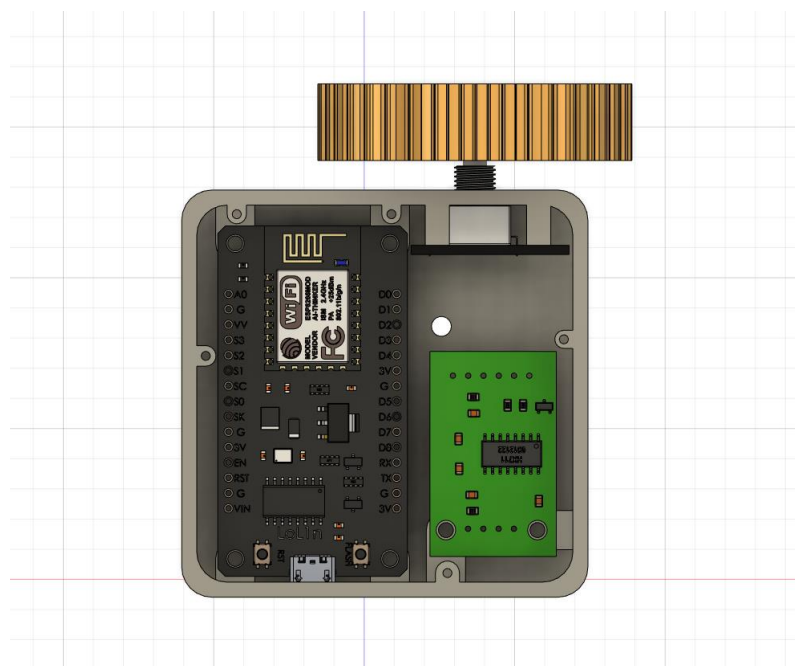


Fig 4.9 cross sectional view of sensor box consisting of M274 encoder, 5kg load cell hx711 module

4.2.1. Final result of Digital twin of screw jack

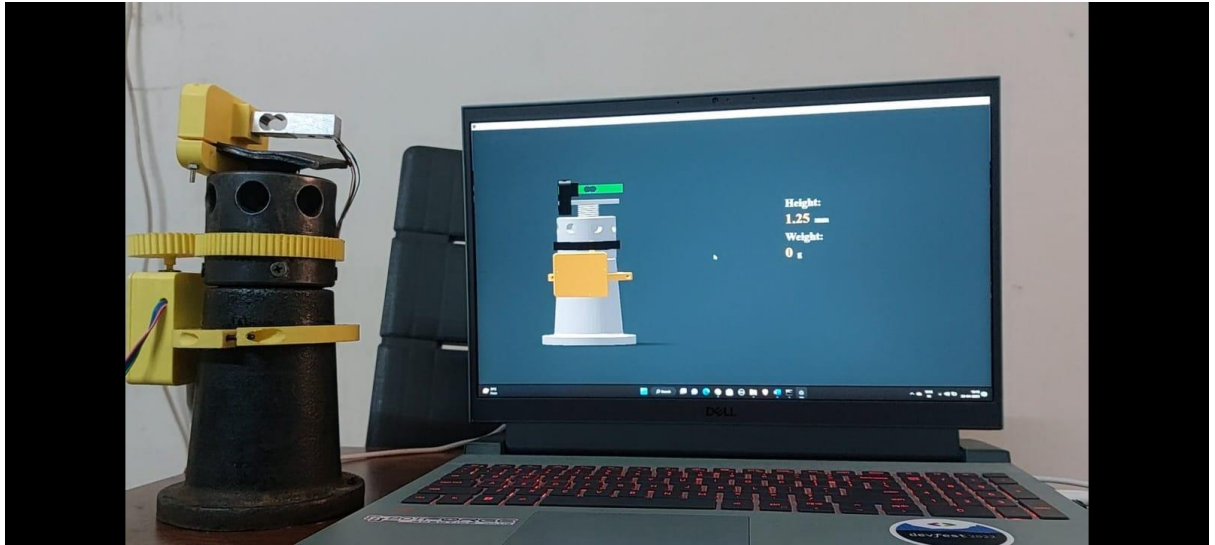


Fig 4.10 Experimental setup of Physical screw jack and its digital twin in ideal case

Initial Weight on Screw jack : 0 gm

Initial height of screw jack : 1.25 mm

Colour of the Load cell : Green

Load Acting : 0 N

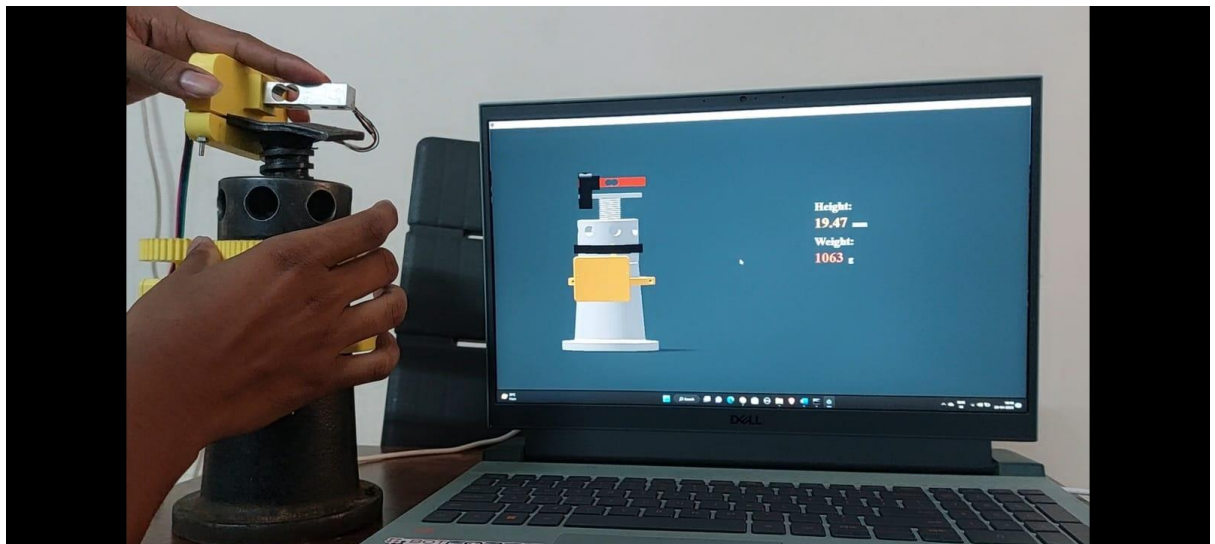


Fig 4.11 Experimental setup of Physical screw jack and its digital twin when load is acting

Weight on screw jack : 1063 gm

Height of Screw jack : 19.47 mm

Colour of the Load cell : Red



CHAPTER 5

CONCLUSION



5. CONCLUSION

Digital twin can be created in numerous ways but the accuracy of the twin created will vary from method to method. Through the 1st method using 'Processing application' the model obtained was not so accurate. It is not recommended to create digital twin for complex structures like mechanical components through this method of processing application.

Whereas it is highly recommended to use web application method using three.js to create a digital twin of any object. Fusion 360 is a very sophisticated and highly reliable 3d modelling software which is used to create 3d models of all types ranging from minute nut and bolt to complex mechanical component like screw jack. Digital twin technology has proven to be a valuable tool in solving mechanical problems related to real-time analysis and detecting failures of mechanical components.

By creating a virtual replica of a physical system or component, engineers can analyze its behaviour and predict its performance under different conditions. Real-time analysis is essential in mechanical systems to ensure optimal performance and prevent failures. Digital twin technology enables engineers to monitor and analyze the behaviour of mechanical systems in real-time, allowing for early detection of potential problems. This early detection can lead to faster and more efficient maintenance, reducing downtime and improving overall system reliability. In addition, digital twin technology can be used to simulate different scenarios and test the performance of mechanical systems under various conditions. This allows engineers to optimize the design and operation of mechanical systems, leading to better performance, increased efficiency, and reduced costs. Overall, digital twin technology has significant potential in solving mechanical problems related to real-time analysis and detecting failures of mechanical components. As the technology continues to evolve and improve, it is likely to become an even more valuable tool for engineers working in the mechanical engineering field.

This is the State of the art technology which is currently utilized by Aerospace and Automobile industries like General Aerospace, IBM etc., We have successfully developed new methodology to implement this digital twin technology in mechanical components.



REFERENCES

1. Michael G. Kapteyn, Karen E. Willcox : Design of Digital Twin Sensing Strategies Via Predictive Modeling and Interpretable Machine Learning // amses-journal.springeropen.com/articles/10.1186/s40323-020-00147-4
2. Xiaonan Lai, Xiwang He, Shuo Wang : Building a Lightweight Digital Twin of a Crane Boom for Structural Safety Monitoring Based on a Multifidelity Surrogate Model // digitaltwin1.org/articles/1-12
3. Xiaonan Lai, Xiwang He, Yong Pang : A Scalable Digital Twin Framework Based on a Novel Adaptive Ensemble Surrogate Model // researchgate.net/publication/337019778
4. Ricard Sanchis, Salvador Cardona, Jordi Martinez : Determination of the Vertical Vibration of a Ballasted Railway Track to Be Used in the Experimental Detection of Wheel Flats in Metropolitan Railways // mdpi.com/journal/applsci/special_issues
5. Aidan Fuller, , Zhong Fan, Charles Day : Digital Twin: Enabling Technologies, Challenges and Open Research
6. Brian Danchilla: Three.js Framework, Springer, 2012
7. Rushikesh Kore, Hansel D'silva, Augmented Reality clothes fitting web app using Three.js & Posenet // researchgate.net/publication/302306483_Threejs_Framework
8. Aashish Gyawali, Modeling & Structural Analysis of Screw Jack // www.researchgate.net/publication/361143069