Kaunas University of Technology

Faculty of Mechanical Engineering and Design

Design and Investigation of a 6-DOF Platform for Educational Applications

Master's Final Degree Project

Sumedh Somashekhar Rajavamshi

Project author

assoc. prof. dr. Darius Eidukynas

Supervisor

Kaunas, 2026

Kaunas University of Technology
Faculty of Mechanical Engineering and Design

Design and Investigation of a 6-DOF Platform for Educational Applications

Master's Final Degree Project
Mechanical Engineering (6211EX009)

Sumedh Somashekhar Rajavamshi
Project author

assoc. prof. dr. Darius Eidukynas
Supervisor

dr. Darius Mazeika
Reviewer

Kaunas, 2026

Kaunas University of Technology

Faculty of Mechanical Engineering and Design

Sumedh Somashekhar Rajavamshi

Design and Investigation of a 6-DOF Platform for Educational Applications

Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;

2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;

3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;

4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Sumedh Somashekhar Rajavamshi

Confirmed electronically

Kaunas University of Technology

Faculty of Mechanical Engineering and Design

1. Task of the Master's Final Degree Project

Given to the student – Sumedh Somashekhar Rajavamshi

1. Topic of the project

Design and Investigation of a 6-DOF Platform for Educational Applications

(In English)

6 laisvės laipsnių platformos, skirtos edukacinėms reikmėms, projektavimas ir tyrimas

(In Lithuanian)

2. Aim and tasks of the project

Aim: To design and investigate a 6DOF Stewart platform for educational applications by developing a lab work for bachelor level students.
Tasks:
1. To perform literature analysis on existing 6DOF Stewart platforms, their inverse kinematics, and existing teaching tools which make use of similar mechatronic systems.
2. To design Stewart platform mechanical part.
3. To perform initial simulations and formulate a mathematical model for the Stewart Platform.
4. To develop a final prototype of the Stewart Platform.
5. To develop the lab work structure which is intended for educational purpose.

3. Main requirements and conditions

The length, width and height of the designed platform should be no less than 100mm, 100mm and 200mm and no more than 700mm, 500mm, and 500mm, respectively.

4. Additional requirements and conditions for the project, report and appendices

"Not applicable"

Project author	Sumedh Somashekhar Rajavamshi	Sumedh	05/02/2026
	(Name, Surname)	*(Signature)*	*(Date)*
Supervisor	assoc. prof. dr. Darius Eidukynas		
	(Name, Surname)	*(Signature)*	*(Date)*
Head of study field programs	assoc, prof. dr. Kęstutis Pilkauskas		
	(Name, Surname)	*(Signature)*	*(Date)*

Sumedh Somashekhar Rajavamshi. Design and Investigation of a 6-DOF Platform for Educational Applications. Master's Final Degree Project / supervisor Assoc. Prof. Dr Darius Eidukynas; Faculty of Mechanical Engineering and Design, Kaunas University of Technology.

Study field and area (study field group): Mechanical Engineering, Engineering Sciences

Keywords: Linkage, Actuator, Pose, Degree-of-freedom, Transformation matrix

Kaunas, 2026. 62 p.

Summary

Recent advancements in the field of mechatronics have increased the importance of engineers having not only a strong theoretical foundation but also a strong hands-on project experience to complement it. Modern engineering graduates are increasingly expected to possess practical competencies in prototyping and rapid problem-solving, which are typically developed through hands-on learning. Mechatronics engineering is a highly multidisciplinary field due to the synergistic influence of various sectors, such as mechanical engineering, electronics, computer programming, and control systems engineering. However, engineering education in the past has been mainly theory-oriented, and industry-relevant practical skills have received very little attention in that field. But recent educational developments have seen a steady increase in universities adapting modules that are more practice-oriented.

This thesis investigates existing educational platforms and teaching tools used for practical mechatronics education, focusing on identifying effective methods to incorporate hands-on multidisciplinary learning. A comprehensive literature analysis was conducted, focusing on currently available educational platforms, robotic teaching systems and laboratory-oriented learning tools being used in engineering education. The literature review examined various educational solutions, including dobotic manipulators, modular educational kits, programmable mechatronic systems and commercially available platforms such as LEGO Mindstorms, etc. The analysis revealed the importance of accessibility, modularity and interdisciplinary integration of modern engineering educational platforms.

Based on the findings of the literature analysis, a six-degree-of-freedom Stewart Platform or hexapod was chosen as the practical component of this thesis. The Stewart Platform was chosen for its highly multidisciplinary nature and its ability to integrate various mechatronics concepts into a single system. A Hexapod was hence designed and developed as an educational tool for bachelor-level students. The design process included the mechanical structure development, mathematical model formulation, servo motor integration, electronics system implementation and software development required for the control of the system. In addition to the physical development of the platform, a laboratory work structure was proposed that employs the developed hexapod across four distinct exercises focusing on mechanics, electronics, programming, and control systems engineering. The developed framework enables students to apply theoretical concepts in a practical environment while promoting hands-on learning and problem-solving skills. Furthermore, with a total implementation cost of approximately 140EUR, the developed platform provides an affordable educational solution that can be adapted to university-level mechatronics and robotics laboratories.

In conclusion, this thesis presents a literature review of existing educational mechatronics and robotics platforms, along with the physical development and implementation of an educational Stewart Platform, along with the accompanying laboratory framework. The developed system showcases the potential of an integrated project-based educational tool for mechatronics education.

Sumedh Somashekhar Rajavamshi. 6 laisvės laipsnių platformos, skirtos edukacinėms reikmėms, projektavimas ir tyrimas. Magistro baigiamasis projektas / vadovas doc. dr. Darius Eidukynas; Kauno technologijos universitetas, Mechanikos inžinerijos ir dizaino fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Mechanikos inžinerija, Inžinerijos moklslai.

Reikšminiai žodžiai: svirtinė sistema, Actuator, Pose, Degree-of-freedom, Transformacijos matrica.

Kaunas, 2026. 62 p.

Santrauka

Naujausi mechatronikos srities pasiekimai padidino inžinierių, turinčių ne tik tvirtą teorinį pagrindą, bet ir didelę praktinę projektų patirtį, kuri jį papildytų, svarbą. Šiuolaikiniai inžinerijos absolventai vis dažniau tikisi praktinių prototipų kūrimo ir greito problemų sprendimo kompetencijų, kurios paprastai ugdomos praktinio mokymosi būdu. Mechatronikos inžinerija yra labai daugiadisciplininė sritis dėl įvairių sektorių, tokių kaip mechanikos inžinerija, elektronika, kompiuterių programavimas ir valdymo sistemų inžinerija, sinerginės įtakos. Tačiau inžinerinis išsilavinimas praeityje daugiausia buvo orientuotas į teoriją, o su pramone susijusiems praktiniams įgūdžiams šioje srityje buvo skiriama labai mažai dėmesio. Tačiau pastaruoju metu švietimo srityje nuolat daugėja universitetų, pritaikančių labiau į praktiką orientuotus modulius.

Šiame darbe nagrinėjamos esamos edukacinės platformos ir mokymo priemonės, naudojamos praktiniame mechatronikos mokyme, daugiausia dėmesio skiriant veiksmingų metodų, kaip integruoti praktinį daugiadisciplinį mokymąsi, nustatymui. Buvo atlikta išsami literatūros analizė, daugiausia dėmesio skiriant šiuo metu prieinamoms edukacinėms platformoms, robotų mokymo sistemoms ir laboratoriniams mokymosi įrankiams, naudojamiems inžinerijos mokyme. Literatūros apžvalgoje nagrinėjami įvairūs edukaciniai sprendimai, įskaitant dobotinius manipuliatorius, modulinius mokymo rinkinius, programuojamas mechatronines sistemas ir komerciškai prieinamas platformas, tokias kaip LEGO Mindstorms ir kt. Analizė atskleidė šiuolaikinių inžinerijos mokymo platformų prieinamumo, moduliškumo ir tarpdisciplininės integracijos svarbą.

Remiantis literatūros analizės išvadomis, praktiniu šio darbo komponentu buvo pasirinkta šešių laisvės laipsnių Stewart platforma arba heksapodas. Todėl heksapodas buvo suprojektuotas ir sukurtas kaip edukacinė priemonė bakalauro lygio studentams. Projektavimo procesas apėmė mechaninės struktūros kūrimą, matematinio modelio formulavimą, servo variklių integravimą, elektronikos sistemos įdiegimą ir programinės įrangos, reikalingos sistemos valdymui, kūrimą. Be fizinio platformos kūrimo, buvo pasiūlyta laboratorinio darbo struktūra, kurioje sukurtas heksapodas naudojamas keturiuose skirtinguose pratimuose, daugiausia dėmesio skiriant mechanikai, elektronikai, programavimui ir valdymo sistemų inžinerijai. Be to, sukurta platforma, kurios bendros įdiegimo išlaidos yra maždaug 140 EUR, suteikia prieinamą edukacinį sprendimą, kurį galima pritaikyti universitetinio lygio mechatronikos ir robotikos laboratorijoms.

Apibendrinant, šiame darbe pateikiama esamų edukacinių mechatronikos ir robotikos platformų literatūros apžvalga, taip pat edukacinės Stewart platformos ir susijusios laboratorinės sistemos sukūrimas ir įdiegimas. Sukurta sistema demonstruoja integruotos projektinės edukacinės priemonės, skirtos mechatronikos mokymui, potencialą.

Table of contents

List of figures

List of tables

List of abbreviations and terms

Abbreviations:

UART – Universal Asynchronous Receiver-Transmitter;

SPI – Serial Peripheral Interface;

I2C – Inter-Integrated Circuit;

IMU – Inertial Measurement Unit;

UJC – Universal Joint Coupling;

FDM – Fused Deposition Modelling;

GUI – Graphical User Interface;

PLA – Polylactic Acid;

PBL – Project-Based Learning;

DOF – Degree of freedom;

Pose – Position and Orientation.

Introduction

The rise of mechatronic systems in today's world has greatly influenced the education which is needed for it. There is an ever-growing need for engineering education to shift from only teaching abstract theory toward a more hybrid approach, which effectively merges theory with practical, real-world applications. This rise of mechatronic systems in sectors such as aeronaughtics, automotive and medicine has effectively increased the demand for engineers who not only possess a good theoretical background, but also for those who have a strong hands-on experience working with machines and equipment. The market now demands an engineer who can distinguish the difference between theoretical calculations and the physical limitations of a system. This rising demand for engineers with practical and industrial level experience has motivated universities all around the world to incorporate activities, lab works and assignments which use learning philosophies such as Project Based Learning (PBL), experiential learning and hands-on learning into their curricula. These teaching philosophies work by encouraging the student to build and develop a project of their own while remaining within a certain design constraint. There are several mechatronic and robotic systems available in today's industry, which can be used for educating these students and providing them with the right skills necessary for today‘s industry.

One such mechatronic system is the Stewart Gough Platform or the Hexapod. This is a parallel manipulator robot which is widely used in various industries, ranging all the way from aerospace to medicine. Unlike its serial manipulator counterpart, the parallel manipulator is made of a fixed base and a moving platform, which are attached with the help of six independently operated legs which can change their lengths. These legs are usually formed by using linear actuators or rotary servo motors. This enables a six Degree-of-Freedom (6DOF) motion of the top platform, allowing it to be moved and oriented to any desired position. These machines offer superior stiffness and rigidity while providing a really high level of precision and stability thanks to their parallel design approach. The Stewart Platform would be a great teaching tool for students as it has a perfect blend of mechanics, electronics, computer programming and control systems engineering. However, the Stewart platforms used in today‘s industry can cost anything between tens and a hundred thousand euros, making them inaccessible for most of the universities worldwide.

This project aims to investigate any existing low-cost teaching platforms which are providing a hands-on learning experience to students in the field of mechatronics engineering. Additionally, this project seeks to design and develop a 6DOF Stewart Platform, which will be used in four lab exercises, each focusing on one discipline of mechatronics engineering, effectively creating an interactive learning experience for students, which teaches them both theory and a practical implementation of that theory.

Aim: Design and investigate a 6DOF Stewart platform for educational purposes by developing a lab work for bachelor level students.
Tasks:
1. To perform literature analysis on existing 6DOF Stewart platforms, their inverse kinematics, and existing teaching tools which make use of similar mechatronic systems.
2. To design Stewart platform mechanical part.
3. To perform initial simulations and formulate a mathematical model for the Stewart Platform.
4. To develop a final prototype of the Stewart Platform
5. To develop the lab work structure which is intended for educational purpose.

1. Literature analysis

In order to explore solutions, the problem statement first needs to be established. The problem is that it is very difficult to provide mechatronics students with the hands-on experience necessary to truly understand the different branches of mechatronics, such as mechanics, electronics, control systems, and computer science. A thorough analysis of several existing systems was performed in order to identify the limitations in the education of mechatronics engineering.

1.1. Engineering Education in Mechatronics

Engineering education in mechatronics was created to provide a systemic understanding of engineering by bridging traditional disciplines such as mechanical, electrical, electronics, and computer engineering [1]. The objective of mechatronics education is to foster a multidisciplinary understanding which allows for a balance of simultaneous systems optimising and information processing [1–3].

Mechatronics education in the modern world emphasises moving away from purely theoretical instruction toward more hands-on, integrated learning modules. The two primary modules in this field are 'project-based learning (PBL)' and 'learning in project implementation' or 'experiential and hands-on learning' [4].
These models form the foundation of modern mechatronics education.

1.1.1. Multidisciplinary nature of mechatronics

Mechatronics is defined as a multidisciplinary field of engineering that integrates technologies from mechanical, electrical, electronic, and computer science to create smarter devices and machines [3]. It essentially brings all these fields together to optimise the functionality, performance and reliability of modern engineering systems. Some key characteristics of mechatronics include:

a. Synergistic design approach in mechatronic design is not just the assembly of separate components to form a system, but rather a more iterative and integrated process, which follows a synergistic design approach where all the above-mentioned disciplines work in tandem to form a cost-effective and optimal balance between mechanical structures and their overall control [3, 4]. Fig. 1. shows a visual representation of this synergistic design approach in mechatronics

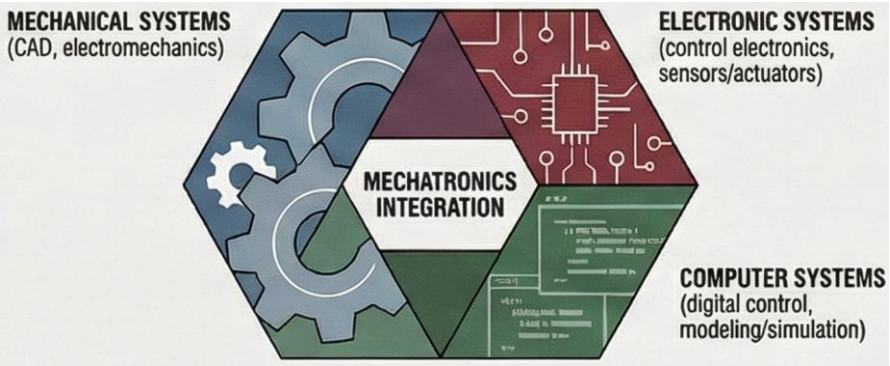


Fig. 1. The fusion of three engineering disciplines to form mechatronics

b. Control systems engineering is considered the 'heart' of mechatronic systems. This involves process control for a system which enables the system to have intelligence, precision and

autonomy. Mechatronics itself is a multidisciplinary field which integrates mechanical systems, electronics, computer science and control theory. Control systems are what coordinate and regulate all these components into a coherent, functional and smart whole [3].

The core role of control systems is to turn hardware into intelligent systems. A mechatronics system without control systems would just be a collection of parts. Control systems provide decision-making logic, real-time response to changing inputs and autonomy. For example, a motor just spins, but a motor with a feedback controller can maintain a particular speed, position or even torque with precision. Fig. 2. shows the schematic of a typical control loop

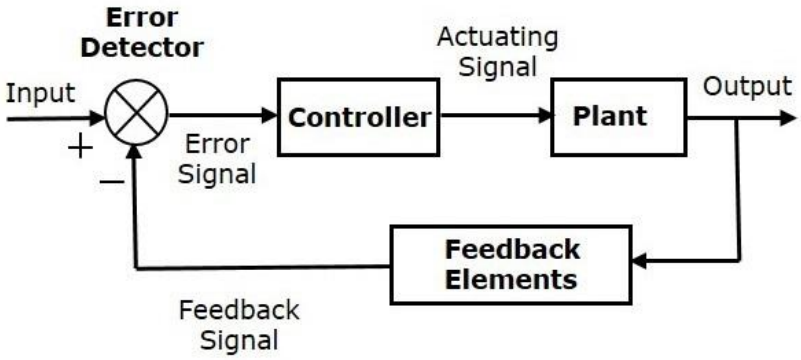


Fig. 2. Components of a control loop

c. Most mechatronic systems rely on some level of informatics, which is the science of processing data for storing and retrieving, also called Information and Communication Technology (ICT). A good mechatronic system is determined by the efficiency of its information being represented, managed and retrieved [3]. Fig. 3. shows the components of ICT.

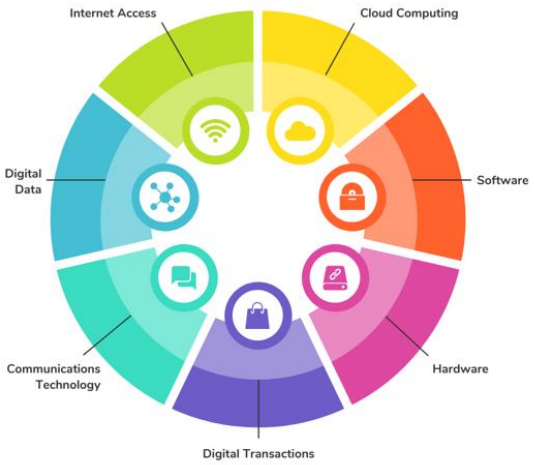


Fig. 3. Components of ICT

ICT provides mechatronics with a computational, communication and data processing backbone, which enables electromechanical systems to become intelligent, connected and

adaptive. While control systems determine how a system behaves, ICT determines how the system thinks, communicates and evolves.

d. Smart mechatronic systems rely on various sensing devices such as encoders, accelerometers, LIDAR sensors and gyroscopes to monitor the system stats. These sensors enable feedback to the controller, which then initiates the actuation of mechanical movements through motors and power electronics [4].

The multidisciplinary nature of mechatronics has given rise to various high-performance and precision systems across multiple sectors, such as industrial robotics, automotive subsystems like antilock braking systems (ABS), countless medical tools, and even common consumer products, such as autofocus cameras [5].

The multidisciplinary nature of mechatronics is often introduced in the form of robotic learning kits and the use of Project-Based Learning (PBL) [5, 6]. With the help of these tools, students can gain a deeper, simultaneous understanding of all the related disciplines of mechatronics, rather than having to study them one by one while solving practical problems related to the real world [6].

1.1.2. Hands-on learning in engineering education

The shift in engineering education in recent times from theoretical instruction toward a more hands-on and experiential learning paradigm is mainly driven by the need to bridge the gap between the academic concepts and the requirements of the industrial sector or the real world [1, 4]. This shift is primarily enabled by the adoption of Project-Based learning (PBL) [5]. It is observed that students understand the concepts with much more efficiency when they are learned through implementation [3].

Hands-on experimentation is crucial for the visualisation of complex technical concepts that could be tricky to grasp through mathematical modelling alone. A few examples of these visualisation applications are:

a. Singularity and workspace analysis is where students can visualise 'singular configurations' where a mechanism loses stiffness or control, also for visualising complex volume shapes of a robot‘s workspace through physical kits. The best example of one such kit is the LEGO Mindstorms, as seen in Fig. 4. [1].

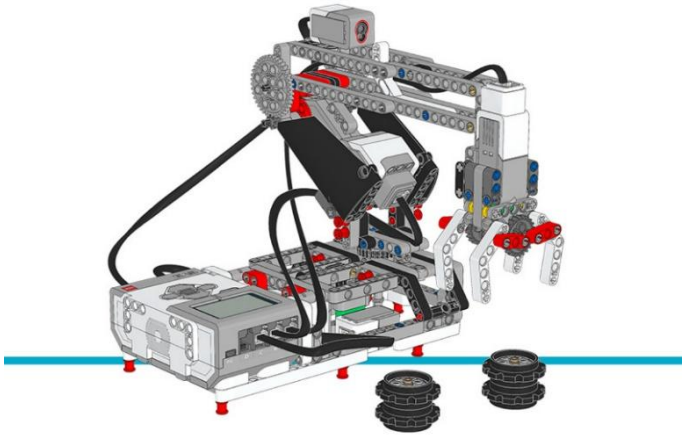


Fig. 4. Visualisation of a robotic arm through LEGO Mindstorms [1]

b. Dynamic response differences through practical experimentation can teach students the difference between a simulated control system and the actual dynamic response of the physical system. The simplest example of this is the varying weight loads affecting the stability of an inverted pendulum. The simulation and the actual response might have some variations due to the real-world limitations, such as system friction and air resistance [5]. Fig. 5. shows the lab setup

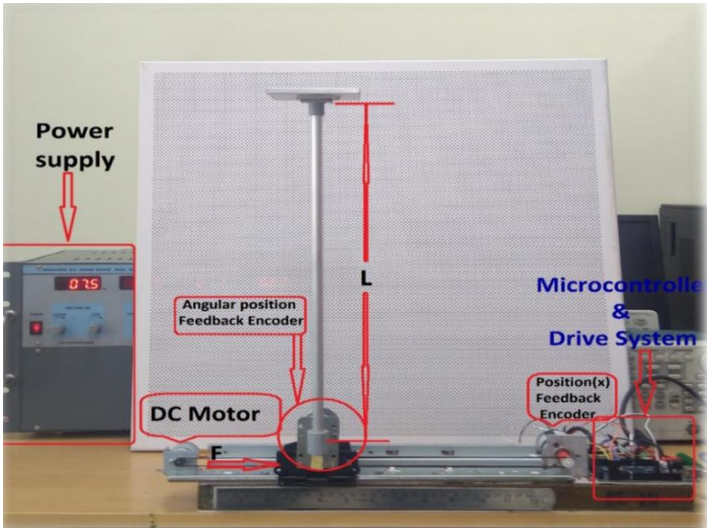


Fig. 5. Set up of an inverted pendulum to investigate the dynamic response difference [5]

c. Concepts of data interpretation and information extraction are critical for understanding the electronics and controls part of mechatronics. Designing and constructing physical circuits, such as filters for sensor signals, provides a solid foundation for those concepts. Digital simulations cannot always provide these insights to students [5].

Apart from technical proficiency, hands-on learning projects can help develop some 'soft' skills and professional behaviours in the students as well, which are highly valued in today‘s industrial sector. These group-based projects can teach students to work in teams, teach them time and task management [1, 2, 4]. The integration of hands-on learning tools in today‘s engineering education has become a necessity for fresh graduates to find suitable jobs, as the industrial sector increasingly demands engineers who can "hit the ground running" with a balance of theoretical understanding and practical knowledge. Some small-scale instrumented devices, such as traction elevators, slot cars and parallel manipulators, are great examples of low-cost, high-fidelity projects that can prepare the students for these real-world systems [7].

1.1.3. Challenges in teaching mechatronics

The pedagogical landscape of mechatronics comes with its own set of challenges because of its interdisciplinary nature, which requires a cohesive integration of various study fields. To provide the students with a complete education curriculum which effectively connects theoretical academic frameworks with the practical industrial applications, educators must overcome several major obstacles [1, 8, 9].

The most significant obstacle is that some of the subject matter is just hard to understand, especially when it comes to kinematics and control theory. It can be tricky to teach the parallel manipulators at the undergraduate level because of the complex math involved in their inverse kinematics, which can

be tricky to grasp sometimes [10]. This complexity is made worse by the need to solve high-parameter math problems. For example, a general serial manipulator has 132 parameters, which are used to describe it. This makes it harder for the students to learn and often requires high computational power [8].

Another key obstacle is financial and resource constraints. This is a major hurdle for practical education, as it limits the accessibility of this discipline in many regions due to the high cost of developing or acquiring industrial-grade machines such as parallel manipulators and robotics laboratories, making mechatronics difficult to teach. This is especially seen in developing nations, where the high cost of these equipments prevent the widespread use of robotics kits despite their versatility. This financial barrier forces people to come up with creative, low-cost alternatives, such as the LEGO Mindstorms, which can, in some way, help bridge the gap between theory and experimentation [1, 10].

Even when the tools and resources are available, teachers find it challenging to prepare theoretical lessons which are relevant to the students' lives by using some real-world examples [1]. A major problem is that theoretical kinematic models don‘t always match up with how the machines actually work. This can occur with things like flexible bases, wrong link lengths, or incorrectly calculated values being entered into a controller. Also, some real-world limitations of low-cost educational prototypes, such as link elasticity or joint clearances, can make them less accurate and reliable, which can be frustrating for students trying to make sense and compare the mathematical model to the real-world prototype [8, 10].

The last set of barriers consists of people and the things in the environment that make it harder to teach the curriculum. More often than not, students don‘t know how to judge their own abilities, which they end up underestimating the complexity of a task and not utilising their time more wisely. This means they would have to work extra hard right before the deadlines [2]. Also, dips in population, like Japan‘s falling birthrate, can make it harder to find good engineers in scientific fields because people tend to move away from them. These logistical problems are important, but are usually not as important as the basic conceptual and economic problems discussed above [2, 5].

1.2. Educational Platforms for Mechatronics Learning

Educational mechatronics platforms are designed to bring together mechanics, electronics and computer science in a way that provides the students with insight into how these areas interact with each other in real engineering systems [1, 3]. Rather than focusing on only theoretical knowledge, such as control theory and kinematics, the focus is shifted toward linking these concepts to a physical and tangible implementation [6, 10, 11]. A wide range of tools is used in this field, from commercial robotic kits like LEGO Mindstorms, ROBIX RCS-6 and BIOLOID Premium, which work on hands-on and modular learning, to open-source platforms like the PlatROB and articulated robotic arms, which enable advanced exploration of topics like AI and autonomous control [12–14].

1.2.1. Robotics platforms used in education

Robotic systems in education work as advanced pedagogical tools which are aimed at teaching skills in automatic control theory, programming and mechatronics [6, 13, 15]. These systems bridge the gap between abstract theoretical formulae and the real world. They make it possible for the students

to "learn by doing", where their need for theoretical knowledge grows along with the complexity of facing real-world problems. Modularity and open-source accessibility have become key characteristics in modern educational robotics, which help reduce the financial barriers and promote system integration skills [6, 14].

There are several open-source robotics kits available right now. One such kit is the PlatROB. This is an open-source, low-cost platform comprising four 3D-printable modules. An Ackermann Drive Module (ADM), used for learning automotive dynamics. An Omnidirectional/Differential Drive Module (ODM/DDM), used for manoeuvrability studies. A Control and Processing Module (CPM) that utilises a Jetson Nano for AI and computer vision projects. And a 4-DOF Articulated Manipulation Module (AMM), used for learning kinematics. These modules can be interconnected as well [14]. Fig. 6. shows the four primary PlatROB modules

Fig. 6. PlatROB robotics learning kits [14]

There are also some modular ecosystems like Mbot and Flower Bots, which provide scalable learning environments that grow with the learner, transitioning from basic mechatronics to advanced ROS-based (Robotic Operating System) autonomy. There are also some low-cost robotics kits, such as Mona, HeRo, and Thymio, which provide an entry-level accessibility often used from kindergarten through university, mainly to foster engagement in scientific fields [14].

There is also a sector in robotics teaching kits which focuses on articulated manipulators and serial arms that teach kinematics, dynamics and autonomous object manipulation. The articulated serial manipulator, as seen in Fig. 7., is a vertical articulated arm which is equipped with optical encoders and force-sensing resistors (FSR) to teach path planning and tactile feedback. Industrial training proxy systems like the SCORBOT-ER 4u, AL5A, and XR-4 teach students to implement advanced control strategies, such as Sliding Mode Control (SMC) or the Disturbance Observer Based Control (DOBC), which are often not available on simpler platforms [13].

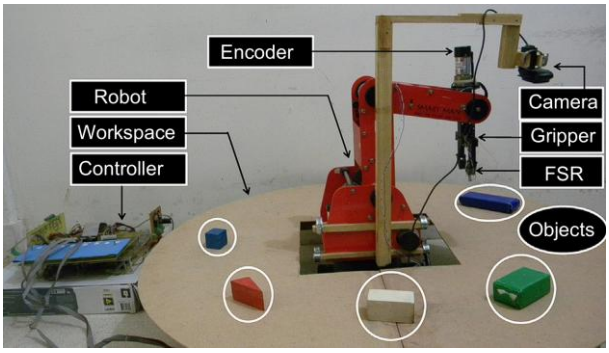


Fig. 7. Articulated serial manipulator teaching kit [13]

1.2.2. Modular mechatronics learning kits

Mechatronics and Robotics are very closely related because of their interdisciplinary nature, but are different in terms of scope and focus. Mechatronics is a field that integrates mechanical systems, electronics, control theory and computing to design and optimise smart machines. Robotics is a specialised subset of mechatronics that specifically focuses on designing, building and programming robotic systems capable of autonomous or semi-autonomous operations. In short, mechatronics is a broader field which combines multiple fields, and robotics applies that foundation to intelligent machines that interact with the physical world. When it comes to mechatronics learning kits, there are a few commercial off-the-shelf (COTS) kits which are often adapted due to their modularity and relatively low entry cost compared to traditional robotics laboratories [1].

LEGO Mindstorms NXT/EV3 is one such platform which is well known for its extreme versatility and utility across multiple academic fronts such as robotics, Computer-Aided Design (CAD), and interdisciplinary extension activities. It comes with a 32-bit Programmable Logic Controller (PLC), interactive geared servo motors with precise rotation and feedback, an array of sensors (ultrasonic, light, sound and touch) and a wide range of mechanical components such as columns, joints, connectors, etc. This creates a significant pedagogical utility for LEGO kits to be able to visualise complex robotic structures and phenomena, such as singular configurations where a mechanism loses stiffness or control. Additionally, in CAD instruction, these kits allow students to manage large-scale projects involving 1000 virtual pieces, being able to simulate machine elements such as gears, springs and pulleys [1]. Fig. 8. shows the variety of LEGO Mindstorms learning kits.

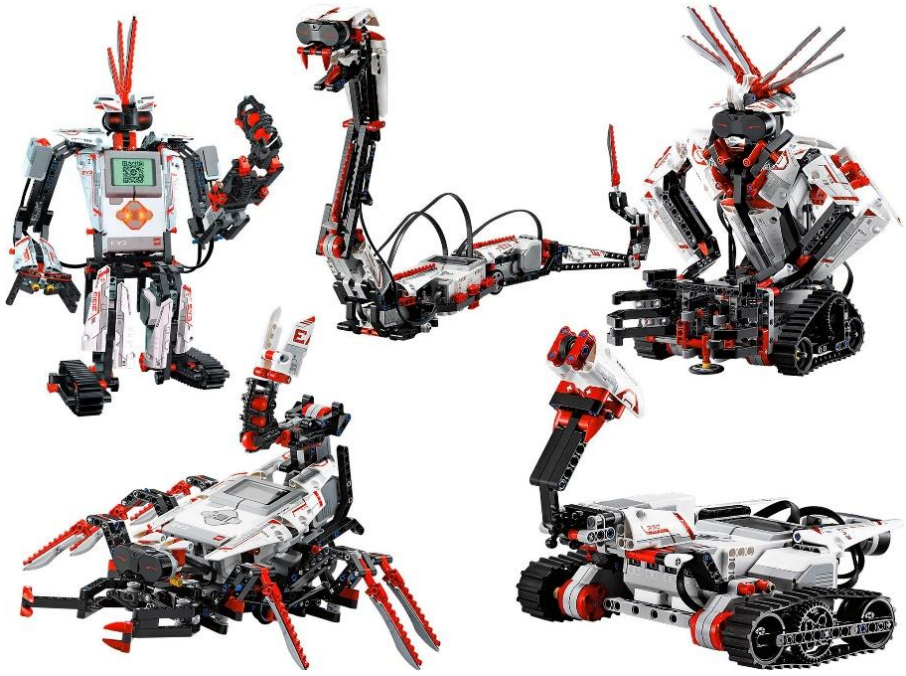


Fig. 8. Versatility of LEGO Mindstorms learning kits [1]

The Bioloid Premium is another learning kit which stands out because of its advanced "Dynamixel" actuators, which are smarter than regular servo motors as they each come with a microcontroller giving them the ability to change and monitor their position, speed, temperature and load. These modules are easy to interconnect, making them highly versatile to create complex robots and mechatronic structures with several moving parts [6]. Fig. 9. shows the variety of Bioloid Premium.

Fig. 9. Versatility of Bioloid Premium [6]

The above-mentioned mechatronic teaching kits are good, but they are still limited to the processing power of the onboard PLCs, the variety of mechanical components and sensors. Also, these kits could be expensive depending on the complexity of the project. To address the limited flexibility and high cost of these commercial offerings, educators have developed some custom modular platforms using standardised parts. These platforms often utilise high-impact aluminium chassis (Lynxmotion A4WD1) and specialised microcontrollers (Pololu Orangutan X2) designed for robots, which come with integrated motor drivers and multiple GPIO ports for integrating sensors such as GPS modules, IR sensors, IMUs, etc. These modular systems often employ a Real-Time Operating System (RTOS), such as FreeRTOS, because of its ability to divide complex software into independent tasks based on priority. These RTOS are open source and often come with a lot of documentation, which makes them easy to use and implement [16].

1.3. Learning Methodologies in Mechatronics Education

Modern mechatronics education utilises a variety of teaching methodologies which focus on bridging the gap between abstract theoretical knowledge and the practical needs and industrial application of that knowledge [3, 5, 6]. These methodologies mainly come in three distinct frameworks. Based on the approach and implementation. The ideal process would be to follow a synergistic blend of these three frameworks when framing a study curriculum. The following frameworks are:

1.3.1. Project-Based Learning (PBL)

Project-Based Learning or PBL is a learner-centred teaching method which revolves around teaching complex, real-world engineering concepts through projects which solve a real-world problem. Inspired by John Dewey's "learning by doing" and David Kolb‘s experiential learning theory, it emphasises learning through collaboration, inquiry, and hands-on practice [17, 18]. PBL projects span a long period of time, typically for a semester or longer, where students get to implement knowledge from multiple disciplines and solve open-ended problems, which do not always have a single correct answer, which means that students take greater responsibility for their learning, managing tasks, identifying knowledge gaps and developing new skills while the instructors act more like facilitators rather than traditional lecturers. This approach ensures a deeper engagement with both the learning process and the underlying engineering concepts.

In Engineering fields such as mechatronics, PBL has proven to be particularly effective in linking theory with practical application, where students are engaged with tasks such as controlling dynamic systems, developing navigation algorithms, or modelling robotic manipulators such as Stewart platforms using tools like MATLAB and Simulink. This process helps strengthen technical understanding and also broaden professional competencies [2, 8].

1.3.2. Experiential and hands-on learning

Another popular framework in mechatronics education is experiential and hands-on learning. This is similar to PBL but differs in the approach. It is a broader learning theory rather than a specific teaching method. Here, the emphasis is less on reaching the final result and more on the learning cycle itself, which involves engaging in an activity, reflecting on it, forming concepts, and applying them to another project. Experiential learning can occur within PBL and also in similar activities, such as laboratory experiments, simulations or short-hand tasks. The duration is determined by the type of task [18].

Educators usually rely on accessible hardware platforms that make experimentation feasible beyond traditional labs to support this form of learning. Portable systems, such as small-scale motor experiments, allow students to test concepts independently, whereas modular systems with 3D printed components and off-the-shelf electronics reduce cost and complexity [19, 20].

1.4. Parallel Manipulators and Stewart Platforms

A parallel manipulator is a closed-loop robotic mechanism where a movable platform is connected to a fixed base by linkages or legs which can change their lengths, effectively manipulating the position of the movable platform based on the lengths of each leg.

1.4.1. Serial vs Parallel Manipulators

Serial and parallel manipulators are both mechatronic systems which are widely used in today‘s manufacturing industry, but they both differ in terms of design, kinematic structures, mechanical properties and application [11, 21].

a. Serial Manipulators are inspired by the upper human limb and consist of an open-loop kinematic chain where linkages are connected sequentially, leading to the end effector. These typically have a much higher degree of freedom compared to parallel manipulators. A key feature of the serial manipulator is the ability to customise the number of DOF according to the application [22]. When it comes to the kinematic configuration, these systems are typically described in joint space, as each link depends on the previous joints and links, because of which there is a risk of lag or hindrance when the end effector tries to reach the final position. However, their forward kinematics is simple to solve compared to the parallel manipulators. Serial Manipulators have high dexterity and fidelity and are easier to design and control, making them suitable for applications such as industrial robotic arms used for welding, assembling and painting. Some serial manipulators are also used in the medical industry as surgical robots [11]. Serial manipulators often tend to have high inertial forces and relatively low stiffness, as they are fixed only on one end. Fig. 10. shows what a typical parallel manipulator looks like in the form of a robotic arm. It is observed that the robotic arm has

multiple components, but the basic understanding is that one end is fixed and the other end moves the combined DOF of all the arms before it.

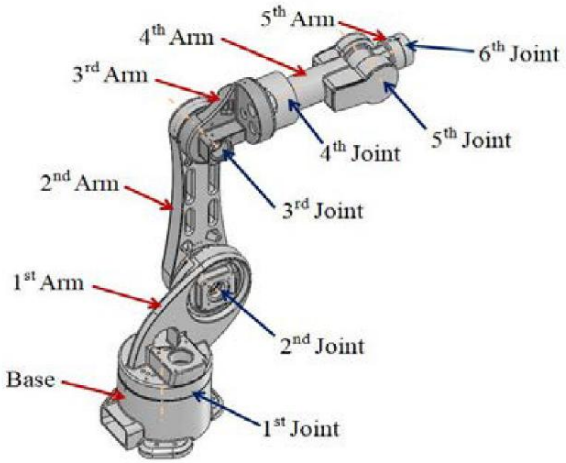


Fig. 10. Serial manipulator and its components [11]

These machines are widely used in the automotive industry as robotic arms that perform tasks such as welding, painting, and moving materials. Another application of serial manipulators is in precision machining.

b. Parallel manipulators, on the other hand, are robotic systems where the end effector is connected to a fixed base by several independent linkages or 'legs' which work in parallel. Unlike the serial manipulator, the linkages do not depend on each other to transmit movement. This structural architecture offers several functional advantages over serial manipulators. Since the structure is a closed loop, it offers far greater rigidity and precision, enabling high-precision positioning in Cartesian space. Since the load is distributed across multiple linkages, the parallel manipulators can handle higher payloads and hence have a high load-to-weight ratio and a low inertia, as most of the weight is concentrated at the base of the platform, which is often fixed or bolted to the ground [11, 21, 23]. Fig. 11. shows what a parallel manipulator looks like in the form of a Stewart Platform.

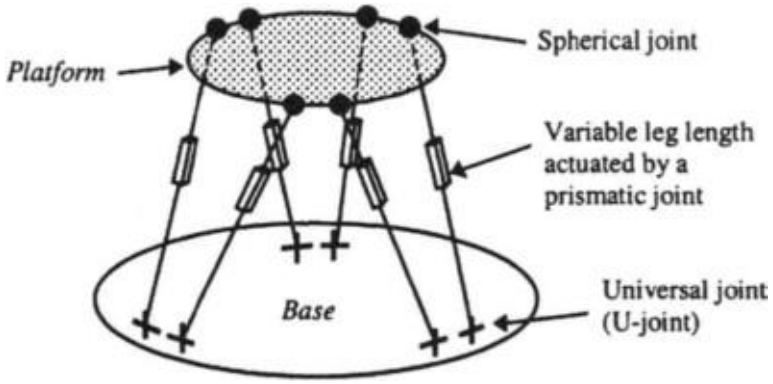


Fig. 11. Parallel manipulator in the form of a Stewart Platform [11]

The Gough-Stewart platform or hexapod typically provides six degrees of freedom, which comprises three rotational and three translational movements. Though the parallel

manipulator has a higher rigidity, precision and payload capacity, it does have a few drawbacks. The mathematical modelling of parallel manipulators comes with an increased complexity compared to serial robots. Also, the parallel manipulators have a more restricted workspace compared to serial robots, often characterised by complex volume shapes determined by the limits of actuators and potential link interference [24].

1.4.2. Structure of Stewart Platform

The Stewart platform, also known as the Gough-Stewart manipulator or hexapod, is a parallel robotic structure which has six degrees of freedom (6-DOF). Its structure consists of a base platform and a movable payload platform. These two platforms are interconnected to each other by six independent linkages, which can change their length‘s axially. These linkages are most commonly referred to as legs of limbs. As mentioned above, the Stewart platform has 6-DOF, which breaks down into three translational (X, Y, Z) and three rotational (roll, pitch, yaw) movements through a coordinated movement of those six linkages. These legs are attached to the base and the moving platform through special joints such as universal, spherical or ball and socket joints [21, 23, 25].

Stewart platforms come in several configurations based on the number of distinct attachment points (joints) on the base and the moving platform.

a. 6-6 Configuration (General Stewart-Gough) is the most general form of the Stewart platform, where all the 12 joints (6 on the base and 6 on the platform) are attached at distinct points on the base and platform, respectively. This configuration provides the most flexibility but is the most complex to solve for mathematically. Fig. 12. shows a Stewart platform with the 6-6 configuration [24, 25].

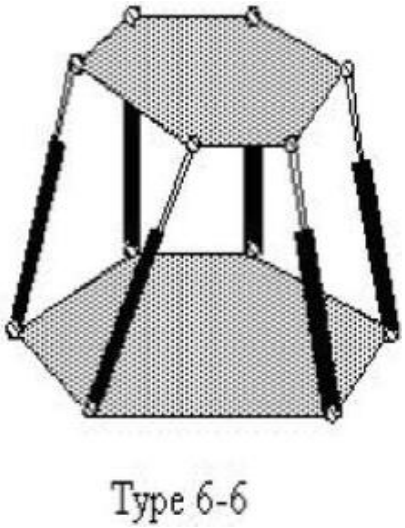


Fig. 12. Stewart Platform with the 6-6 configuration [25]

b. 3-6 Configuration is when the top moving platform has only three joints, but the base still has six distinct joints. The linkages attach to the platform in pairs at three points, as seen in Fig. 13. This effectively forms a triangle on the mobile side. This configuration simplifies the forward kinematics, but the range is also reduced, and in some cases, the DOF is also reduced. Often called a semi-symmetrical design, this configuration balances structural stiffness with simplified mathematical control [24, 25].

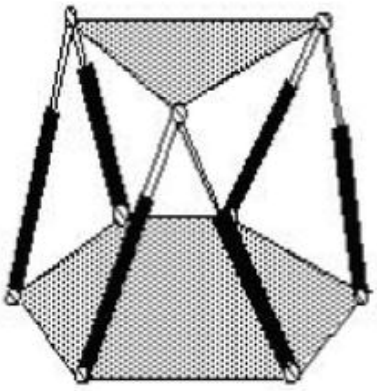


Fig. 13. Stewart platform with the 3-6 configuration [25]

c. 3-3 configuration is when the base and the platform have three joints where the linkages are attached in pairs, forming two triangles, as seen in Fig. 14. This design is highly stable, but the biggest tradeoff is the restricted movement range due to joint interference [24, 25].

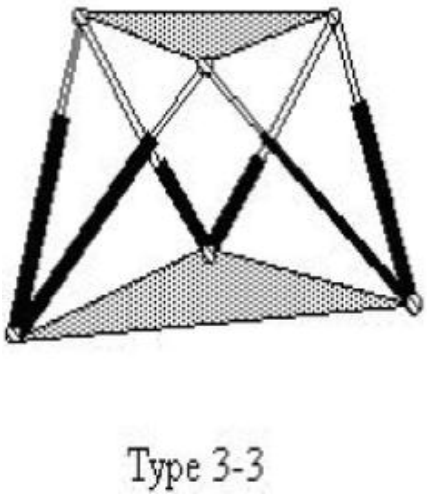


Fig. 14. Stewart platform with a 3-3 configuration [25]

d. Symmetrical Hexagonal platform is the most common configuration, where the joints on the base platform and moving platform are attached in a semi-regular hexagon. This design balances stiffness and a wide range of workspace. Both the base and moving platform are typically shaped like hexagons or circles. Here, the six linkages are arranged in pairs on each side. The base joints are spaced out so that there are three wide gaps and three narrow gaps, and then this pattern is mirrored/offset on the moving platform, where there is a wide gap on the base, and the legs converge to a narrow gap on the top [11]. This can be clearly seen in Fig. 15.

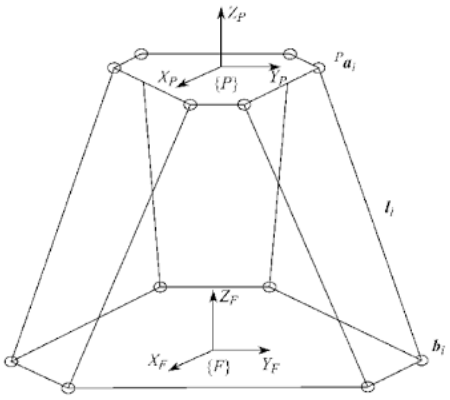


Fig. 15. Schematic of a Stewart platform with a symmetrical hexagonal configuration [11]

Modern high-precision Stewart platforms, particularly those designed for active vibration control, employ some specialised components to enhance performance. Some active components (struts), such as Giant Magnetostrictive Actuators (GMA‘s) as the linkages. These are favoured in space and ultra-precision applications for their ability to offer high force. Large strain and micro-vibration suppression capabilities. Another component is the use of flexible mounts as joints. These eliminate the mechanical issues of friction and backlash, which are inherent in traditional ball joints. These joints provide motion through elastic bending, which is very important for achieving the high levels of precision required in micro-vibration control [26, 27].

1.4.3. Applications of Stewart Platforms

The Stewart platform or Hexapod is a versatile 6-DOF parallel manipulator, which is widely used in various high-precision and high-load sectors due to its high structural rigidity, high force-to-weight ratio and precise positioning capabilities [23, 28]. The different applications of the Stewart platform are:

a. Stewart platforms have been a part of the space industry for a very long time, dating back to their use as flight simulators in the 1660s to train pilots safely [21, 25]. Modern aerospace applications include docking systems on the International Space Station (ISS) and for positioning the mirrors on the James-Webb Space Telescope (JWST).

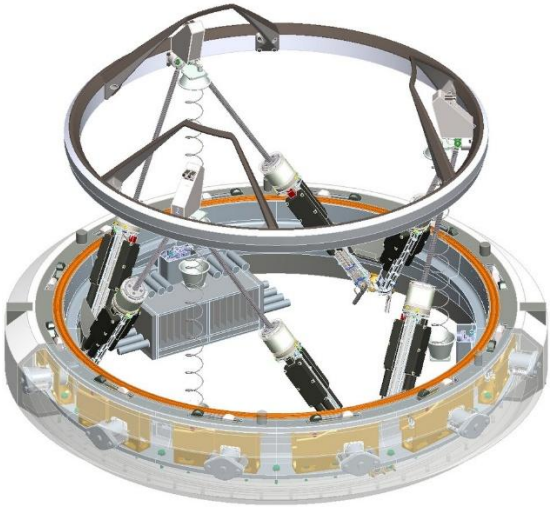


Fig. 16. The Stewart platform used in ISS for docking [21]

At the time of docking, the ISS and the incoming spacecraft are travelling at incredible speeds, and the docking rings on each side must be aligned with incredible precision. Here, the Stewart platforms are very useful. As seen in Fig. 16., the six actuators position the docking ring on the ISS to the exact position of the incoming spacecraft, after which the docking process can be performed by bringing together the docking rings on both sides [28].

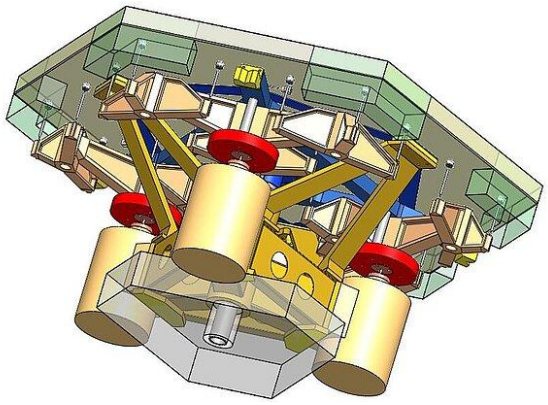


Fig. 17. Single mirror section of the JWST [28]

The James-Webb Space Telescope is the most complex and largest space telescope ever. Developed at NASA, this telescope has a total of 18 mirror segments on it, which reflect the light from space onto a highly sensitive photo sensor. This light is then processed, after which the most detailed photos of space ever are captured. These 18 mirror segments form one giant 6.5 meter long mirror. These mirror segments need to be positioned with a precision of up to a few nanometers to act as one giant mirror. This precise positioning is performed using a hexapod behind each mirror. The design of this hexapod can be seen in Fig. 17., which shows the underside of a single mirror segment [28].

b. Beyond aerospace, these Stewart platforms also serve as the foundation for various simulators. All the high-end vehicle training rigs use these machines to train the drivers by providing the realistic movement of the vehicle. A good example of this application is in the high-performance car racing sector, such as Formula 1. Fig. 18. shows the McLaring Formula 1 racing simulator. These racing simulators need to generate incredibly fast and precise movements to replicate the movements of an actual F1 car [29, 30].

Fig. 18. Racing simulator used by McLaren [29]

Stewart platforms are also used to simulate sea waves for ship motion studies and wave compensation platforms. This is especially useful to generate specific waveforms of water in order to study how the ship or submarine would respond to these movements [21, 29]. These simulators are often very large in size, as seen in Fig. 19.

Fig. 19. Wave simulator for ships and submarines [29]

Lastly, this technology is also applied in 7D theatres. VR games and other interactive gaming systems simulate an experience for the user. The most popular entertainment equipment is the fun rides seen in amusement parks and malls [11, 25]. One such ride can be seen in Fig. 20.

Fig. 20. Rollercoaster simulator in an amusement park [11]

c. Industrial and Manufacturing sector is where the high stiffness of parallel minipulators makes the Stewart platform ideal for heavy-duty industrial tasks, which involve machine tools such as CNC machines and high-speed milling machines, also for some specific robotic operations

such as high-speed pick-and-place robots, robotic cranes for unloading shipping containers and automated assembly lines. These machines are also used in some specific precision systems, such as automobile wheel alignment [21, 25, 31]. Fig. 21. shows a pick-and-place robot, which is a parallel manipulator. This machine is used for high-speed pick-and-place operations in the manufacturing and assembly sector.

Fig. 21. Delta robot used for pick-and-place operations [31]

d. Medical, Research and Testing is another sector with high utility for the Stewart platform. These machines are used in precision medical devices, such as orthopaedic surgical robots and delicate eye surgery tools [25, 32]. Fig. 22. shows an orthopaedic surgical robot which specialises in knee-related surgeries. Here, the moving platform, or end-effector, is the surgical tool, connected to a rigid base by multiple independent kinematic linkages. They are favoured in bone-related procedures as they offer high structural rigidity and precise positioning, which are both highly critical for high-force tasks like bone milling and drilling.

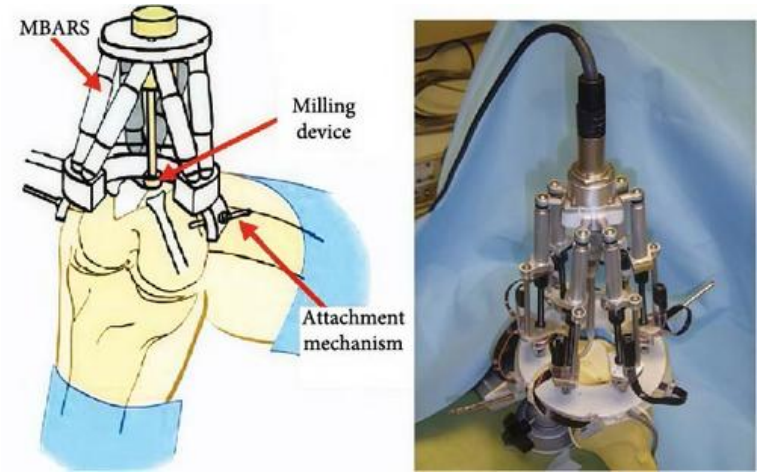


Fig. 22. Stewart platform used for knee-related surgeries [32]

Another medical application of Stewart platforms is in delicate eye surgery. Here, the high rigidity and precision again play a vital role. Fig. 23. shows an eye surgery equipment which ensures that the head does not move during the procedure.

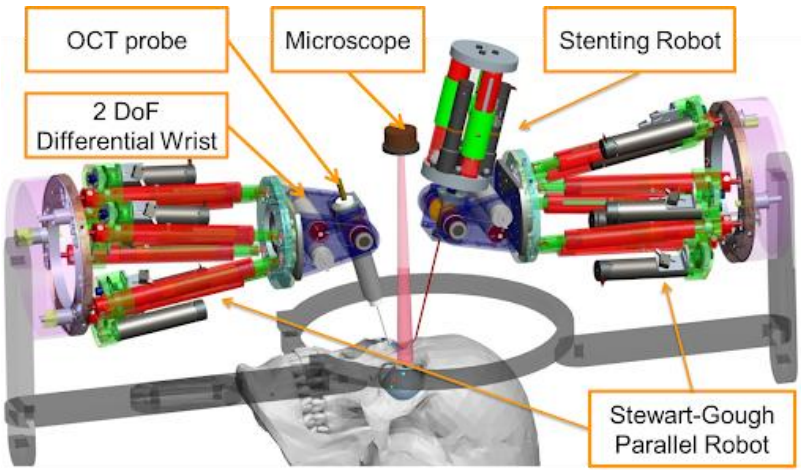


Fig. 23. Stewart platform used for eye surgeries [32]

These machines are also utilised to provide the necessary stability and precision for positioning large radio telescopes, such as Amiba and Nanshan telescopes [25]. Fig. 24. shows a microwave telescope which utilises the Stewart platform at its base for positioning the telescope.

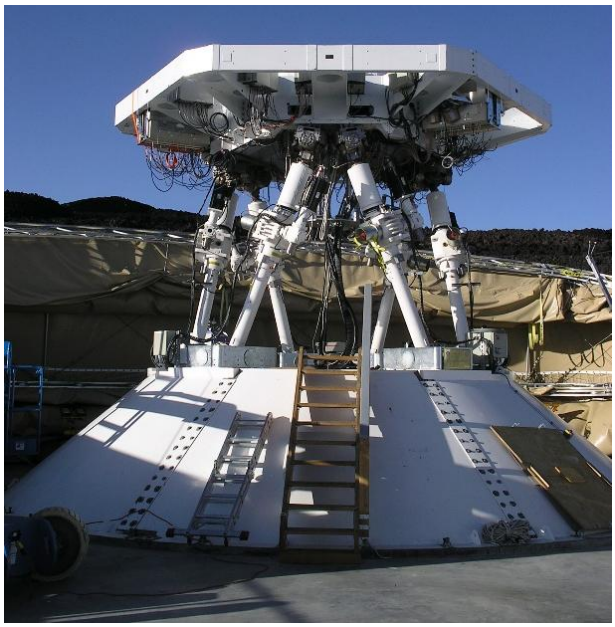


Fig. 24. Microwave telescope [25]

e. Education and Training is where Stewart platforms could be essential in mechatronic educational tools due to their complex kinematics and mathematical modelling. Some academic versions, such as the "Ball and Plate" system, could be utilised to teach students concepts such as control theory, computer vision, and PID tuning in a safe, small-scale laboratory environment [15, 33].

1.5. Stewart Platforms in Educational Applications

In the academic context, these Stewat platforms can be great tools to bridge the gap between abstract mathematical formulae and physical reality. Educators in the modern world employ these parallel manipulators to teach several core engineering concepts to students.

Core mechanical concepts, such as kinematic analysis and mathematical modelling, can be learned through parallel manipulators. Kinematics is mainly divided into two branches. They are forward kinematics (FK) and inverse kinematics (IK). These are geometric methods used in robotics and mechatronics to calculate motion. For example, in a Stewart platform, FK determines the moving platform‘s position given the linkage lengths. In contrast, IK determines the necessary linkage lengths needed for a desired platform position. Students can also learn to tackle singular configurations, where a mechanism loses stiffness or control. This is nearly impossible to comprehend through only theoretical modelling [8, 22].

Control theory is another core concept of mechatronics. The simplified 2-DOF "Ball & Plate" versions are frequently employed to teach PID controller design, which allows students to apply methods like Fiegler-Nicolas tuning to stabilise a physical object in real-time. A more advanced level of student can implement more robust strategies, such as Fuzzy Logic or Adaptive Outcome Control, to handle some system nonlinearities [15, 33].

The process of building a Stewart Platform requires integrating mechanical hardware, such as linear actuators, universal joints, etc, with electronics, such as microcontrollers and sensors and also software like MATLAB or LabVIEW. This develops Hardware-in-The-Loop (HIL), where students get to verify their mathematical model against physical performance [11, 15].

Due to these reasons, a Stewart platform would be a great teaching tool, which will aid the educators to teach mechatronics more efficiently to students, as it provides a bridge between theory and practice. It effectively eliminates the gap between abstract project design and real-world implementation by forcing students to factor in some practical constraints like power saturation, system friction, sensor noise and mechanical backlash [21, 30]. As mentioned before, Stewart platforms are used in all sorts of high-stakes industries such as flight simulators, satellite docking, telescope positioning and surgical robots. Students will be highly motivated to work with such a machine due to the authenticity of the task [7, 25]. Educators can also utilise some low-cost materials like acrylic plates, cheap servo motors and 3D printed parts to make this complex technology accessible in university laboratories.

2. Methodology

After thorough research and analysis of all existing educational systems designed to teach mechatronics students, it was observed that the Stewart Platform (Hexapod) would be a great tool for this purpose. Hence, the design of a primary prototype was carried out, after which it was assessed for possible shortcomings and issues. The final prototype was then designed and developed.

2.1. Overview of the Proposed System

The idea here is to provide a tool which is easy to assemble, wire up, programme and learn from. All while keeping the overall cost of the platform as low as possible. After this, the framework for a lab work is proposed, which is intended to utilise each aspect of the developed Stewart Platform for teaching a specific skill to the students. A total of four aspects are being targeted in this lab work. They are:

a. Mechanical skills: The mechanical aspects can be learned through the process of assembling the platform, taking all the kinematic measurements of the dimensions, learning how the mathematical model works and understanding what Inverse Kinematics is.

b. Electronics skills: The process of understanding which microcontroller is used, the specifications of each servo motor, communication with the feedback sensor and understanding the wiring, etc., teaches the students the electronics aspects of the platform.

c. Control systems: After understanding the mechanical and electronics aspects, the students will progress towards integrating both these aspects through the process of block diagrams, flow charts and schematics. These form the foundation of control systems.

d. Programming: This is the final skill, which brings everything together in a holistic system. The students will learn to implement the mathematical model by writing a programme using Python or MATLAB, which then performs the complex calculations. There is also programming involved on the microcontroller side, which involves establishing communication and control of the servo motors and the feedback sensor. Most basic microcontrollers are usually programmed using the embedded C language.

2.2. Mechanical Design of the Primary Prototype

There are mainly two distinct design approaches when it comes to Stewart Platforms, distinguished by the type of linkage or legs used. These two types are shown in Fig. 25. The type chosen for this project is the one using rotary servo motors, as they are considerably cheaper compared to linear actuators. Linear actuators are known for their superior precision and payload capabilities. The educational application is not critical of these two properties, but rather of the overall cost of the entire system. Hence, the rotary servo motor variant was chosen.

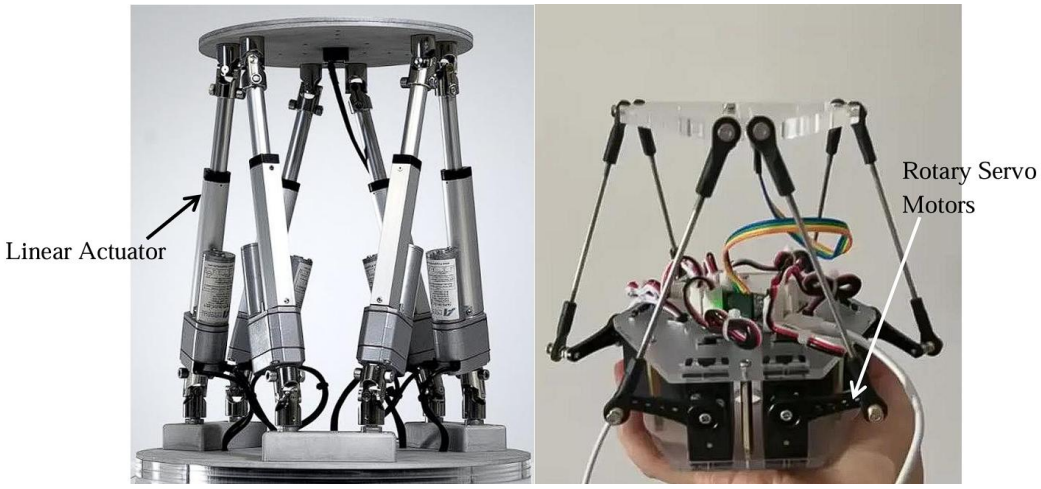


Fig. 25. a) Stewart platform using Linear actuators, b) Stewart platform using servo motors

A primary prototype was designed on SolidWorks, keeping in mind the overall dimensional constraints of the Stewart Platform as seen in Table 1. The mechanical design of the primary prototype can be seen in Fig. 26.

Table 1. Design requirements for the primary prototype

Property	Min, mm	Max, mm
Length	50	250
Width	50	250
Height	100	500

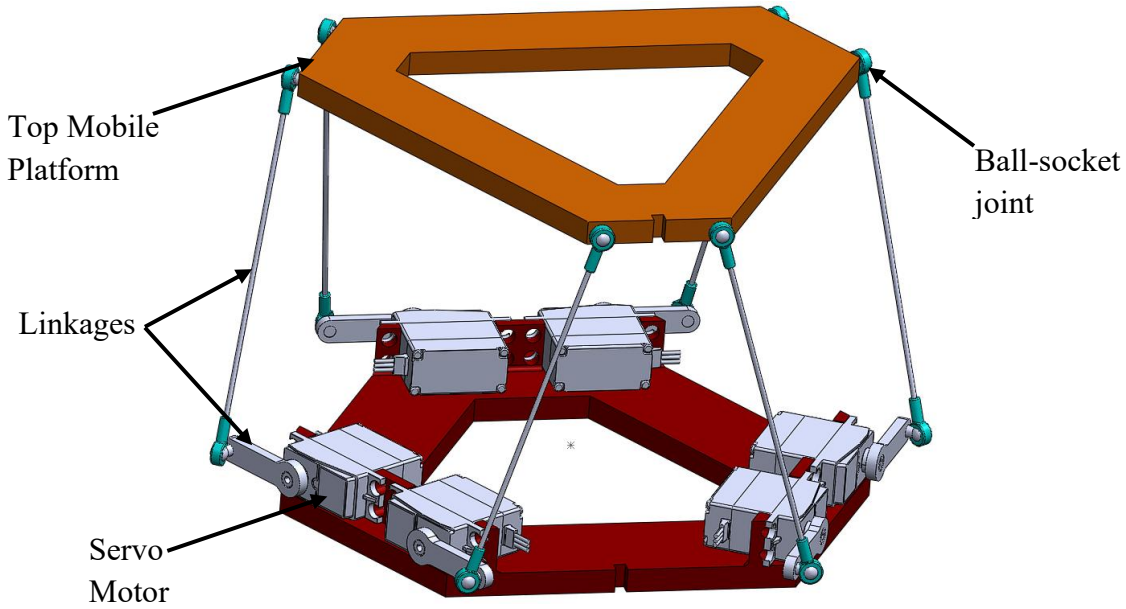


Fig. 26. Design of the primary prototype

This design was primarily intended for initial visualisation and mathematical model formation. A mathematical model for this platform was created by using Inverse kinematics in order to understand the required servo angles for a given position of the mobile platform. The 'Mate Controller' tool in SolidWorks was utilised to set up control of the six servo motor angles.

2.3. Mathematical Modelling of the Stewart Platform

Before formulating a mathematical model for the Stewart Platform, it is important to understand its working principle. In kinematics, a pose is a comprehensive combination of an object‘s (the mobile platform in this case) spatial configuration with respect to a reference frame. The configuration consists of the object‘s six degrees of freedom (DOF), which can be broken down into position (translation in X, Y, Z) and orientation (Pitch, Roll, Yaw). These six DOF values for an object's pose indicate where it is and how it is facing, as illustrated in Fig. 27.

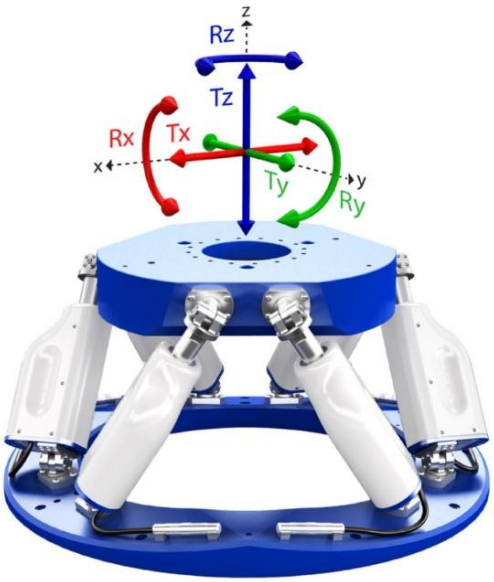


Fig. 27. Illustration of the 6 DOF of a Stewart Platform

The inverse kinematics of a Stewart Platform is the calculation of the leg length given the required pose of the platform. The Stewart platform consists of two frames, the base frame and the platform frame, which are connected with six variable-length legs. With this setup, the platform can be moved in three rotational dimensions and three translational dimensions.

2.3.1. Understanding a single linkage

First, the inverse kinematics of a single linkage needs to be understood. A simplified schematic of a single linkage is shown in Fig. 28.

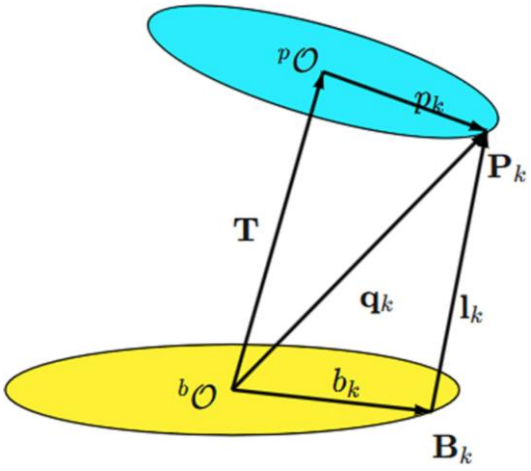


Fig. 28. Schematic of the platform having only one linkage

Here, the point B_k is the point on the base where the lower end of the linkage l_k attaches to the base. This point is fixed and will not move. Hence, its coordinates can be taken from the global coordinate system, assuming that the global origin is located at point bO on the yellow plate from Fig. 28. The base coordinate points are as seen in Equation 1.

$$B_k = \left(x_b, y_b, z_b\right). \tag{1}$$

P_k is the point on the mobile platform (blue plate) where the top end of the linkage l_k is attached to the platform. This point is not fixed, and hence its coordinates will change when the platform's pose changes. Hence, its coordinates need to be taken from the local coordinate system of the blue plate in Fig. 28. The platform coordinates are as seen in Equation 2.

$$P_{k,local} = \left(x_p, y_p, z_p\right). \tag{2}$$

These two points represent the two endpoints of a single linkage. Once this step is repeated five more times, factoring in the six total linkages, these form the coordinate system of the Stewart platform.

2.3.2. Rotation Matrix Formulation

Now, the input parameters for the mathematical model need to be established. As mentioned earlier, there are a total of six input parameters for this platform. It is assumed that in order to move the platform to a global position of T = (X, Y, Z) and with a rotation defined by Pitch (ϕ), Roll (θ) and Yaw (ψ), also known as the Euler angles. So these will become the six inputs.

First, the rotation of the point P_k needs to be calculated with respect to the global coordinate system. Hence, the three given input angles are:

- Pitch (ϕ) – Rotation about the x-axis
- Roll (θ) – Rotation about the y-axis
- Yaw (ψ) – Rotation about the z-axis

To rotate any given point (P_k) in 3D space with reference to an origin point, a single combined rotation matrix has to be used, that factors in these three input Euler angles. This can be seen in Equation 3.

$$R = R_x(\phi) \,.\, R_y(\theta) \,.\, R_z\,(\psi), \tag{3}$$

Where,

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \tag{4}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{5}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

Equations 4, 5 and 6 are equated into Equation 3:

$$R = \begin{bmatrix} \cos\theta\cos\psi & -\cos\theta\sin\psi & \sin\theta \\ \cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi - \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\theta \\ \sin\phi\sin\psi - \cos\phi\sin\theta\cos\psi & \sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix}. \quad (7)$$

Now the new coordinate values of the point P_k are calculated after factoring in the rotation of the platform. Let's say that the coordinates of P_k in matrix form are:

$$P_{k,local} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \quad (8)$$

Now, the dot product of P_k and R is performed, which will solve for the rotated coordinates of P_k with respect to the global coordinate system.

$$P_{rotated} = R \,.\, P_{k,local}. \quad (9)$$

The rotational aspect of the point P_k of the platform has now been solved for. This needs to be repeated five more times to calculate the rotation of all six points on the mobile platform.

2.3.3. Factoring in the Translation

The next aspect is the translational aspect. This is represented by T = (X, Y, X). In simpler terms, it is just the shift of each platform point along the x, y and z axes combined. It is assumed that the desired global position of the platform is:

$$T_{desired} = \begin{bmatrix} x_{desired} \\ y_{desired} \\ z_{desired} \end{bmatrix}. \quad (10)$$

Now, a simple matrix addition of the calculated rotated point coordinates of P_k and the desired global position of the platform $T_{desired}$ is performed. As seen in Equation 11.

$$P_{transformed} = R \,.\, P_{local} + T_{desired}. \quad (11)$$

The rotated and translated platform point P_k have now been solved for a given input of six DOF. Now this whole process needs to be repeated five more times, after which the transformed coordinates of the total six points on the mobile platform will be complete.

2.3.4. Calculation of Leg Lengths

Once transformed coordinates are calculated, the lengths of each linkage can be calculated by calculating the distance between each base point (B_k) and its corresponding transformed platform point ($P_{transformed}$). It is assumed that these two coordinates are the endpoints of a vector L, which can be represented in vector form as seen in Equation 12.

$$\vec{L} = P_{transformed} - B_k. \quad (12)$$

Finally, the linkage length is obtained by calculating the magnitude of the vector L, as represented in Equation 13.

$$l_k = \|L\| = \sqrt{(x_p - x_b)^2 + (y_p - y_b)^2 + (z_p - z_b)^2}. \quad (13)$$

This also needs to be repeated five more times in order to calculate all six linkage lengths for the given input pose.

2.3.5. Calculation of Servo Motor Angles

For Stewart platforms that use linear actuators, the mathematical model would be complete for calculating the linkage lengths. Since the chosen platform uses rotary servo motors, those linkage lengths need to be converted into rotation angles of each servo motor. Fig.29. shows how the linkage is formed in the case of using rotary servo motors.

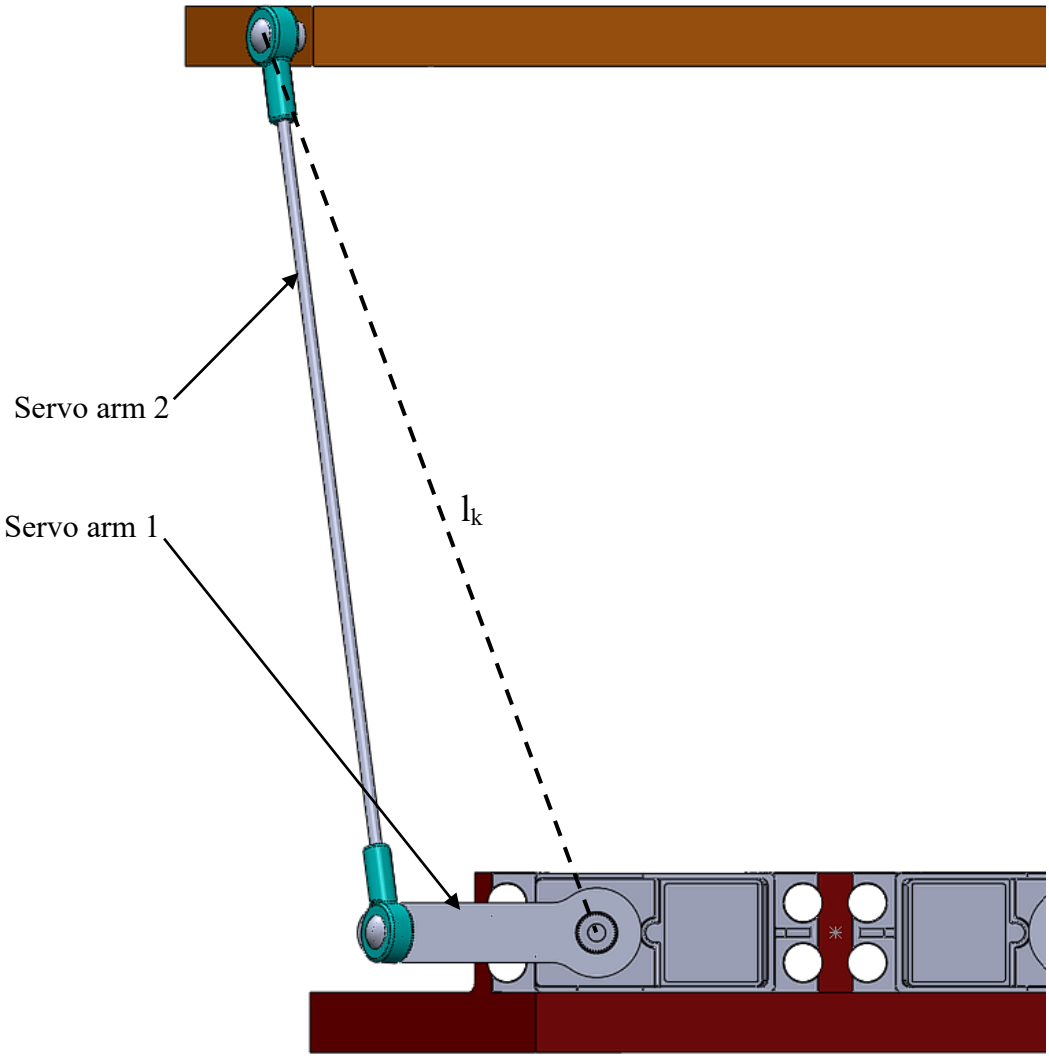


Fig. 29. Linkage formation using servo motor arms

As observed in Fig. 29., the calculated linkage length, l_k is an imaginary line with a varying length. But servo arms 1 and 2 have a fixed length. It is also observed that the angle between these two arms determines the length of l_k. A schematic of this linkage system is seen in Fig. 30.

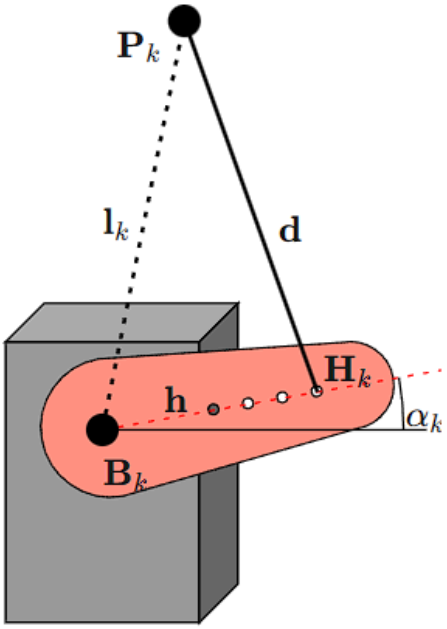


Fig. 30. Schematic of a single servo motor linkage

Here, the line *h* is the servo arm 1 and line *d* is the servo arm 2. Now, the last step in the mathematical model is to calculate the required servo motor rotation angle for a given linkage length, l_k. It is observed that the lines d, h and l_k form a triangle. Using the law of cosines, the required angle can be obtained, as seen in Equation 14.

$$\alpha_k = \cos^{-1}\left(\frac{l_k^2+h^2+d^2}{2l_k h}\right). \tag{14}$$

The mathematical model for this Stewart platform is now complete. This process has to be repeated 5 more times to calculate the required angles for all six servo motors of the Stewart Platform. The calculations have to be performed every time the platform's position changes. This would be impractical if the calculations had to be performed manually every single time. Hence, programming tools such as MATLAB or languages such as Python could be used to create this mathematical model.

2.4. Mechanical Design of the Final Prototype

Since the primary prototype was mostly meant for initial visualisation and mathematical model formation, there were some obvious shortcomings with it when it came to the real-world implementation aspects. Firstly, there was no consideration for the control electronics. The electronic components, such as a microcontroller, a motor driver and a feedback sensor (gyroscope) are needed if the prototype needs to be built in real life. Secondly, the implementation of the above-mentioned mathematical model so that the platform can be controlled using it, through a control algorithm. Lastly, the primary prototype is not very user-friendly, which means that the ease of its assembly and disassembly was not kept in mind while designing it.

The primary prototype was highly useful for the theoretical understanding of Stewart Platforms, but a Stewart Platform which can be built in the physical world is now needed. Hence, the design and development of the second and final prototype was necessary. Fig. 31. shows a CAD model of the final prototype. The dimensional constraints for the final prototype can be seen in Table 2.

Table 2. Design requirements of the final prototype

Property	Min, mm	Max, mm
Length	100	500
Width	100	500
Height	200	700

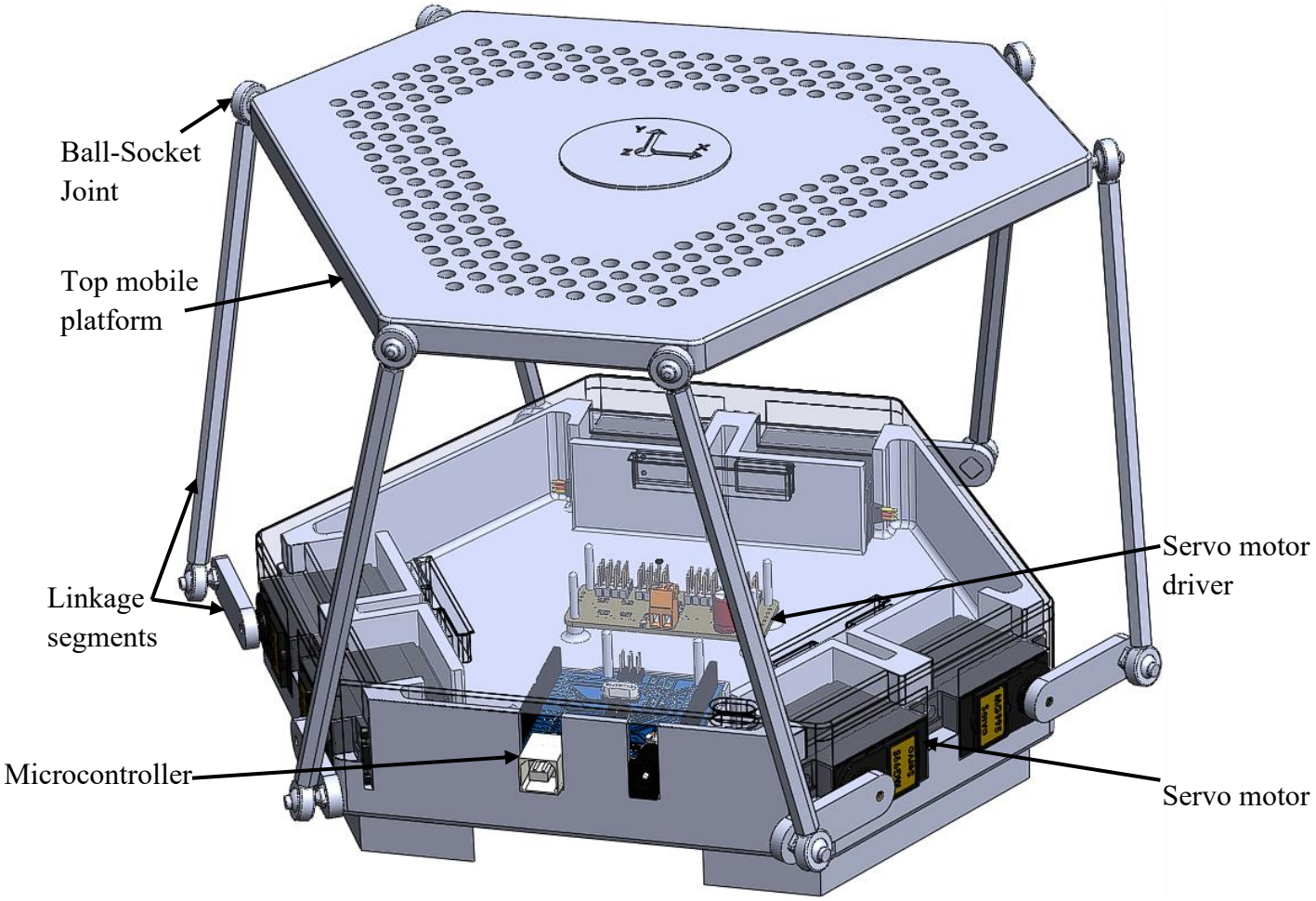


Fig. 31. Design of the final prototype

There are some noticeable differences compared to the primary prototype. The final prototype has a dedicated space for the microcontroller and the servo motor driver. The control algorithm and ease of assembly will be explained in detail in the next sections. To understand the functionality of the different parts of this design, it is broken down into its different subassemblies. There are mainly three subassemblies in this design. First is the base subassembly, second is the linkage subassembly and lastly the movable platform subassembly. The drawings of the platform assembly, along with the drawings of each component, are given in Appendix 3 to Appendix 10

The base subassembly consists of three distinct parts. This subassembly houses all the major electronic components and the servo motors as well. Fig. 32. shows the exploded view of the base subassembly.

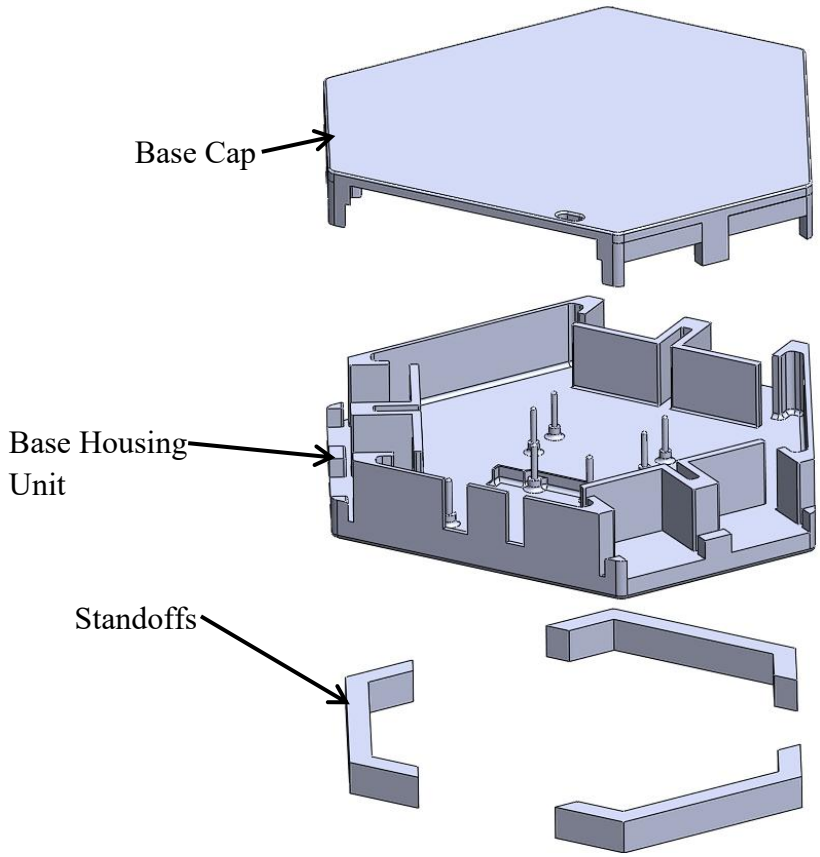


Fig. 32. Components of the base subassembly

Next is the linkage subassembly. Each subassembly consists of two ball-socket joints attached by a rod. The bottom joint connects to the servo arm, and the top joint attaches to the mobile platform. The linkage subassembly and its exploded view can be seen in Fig. 33.

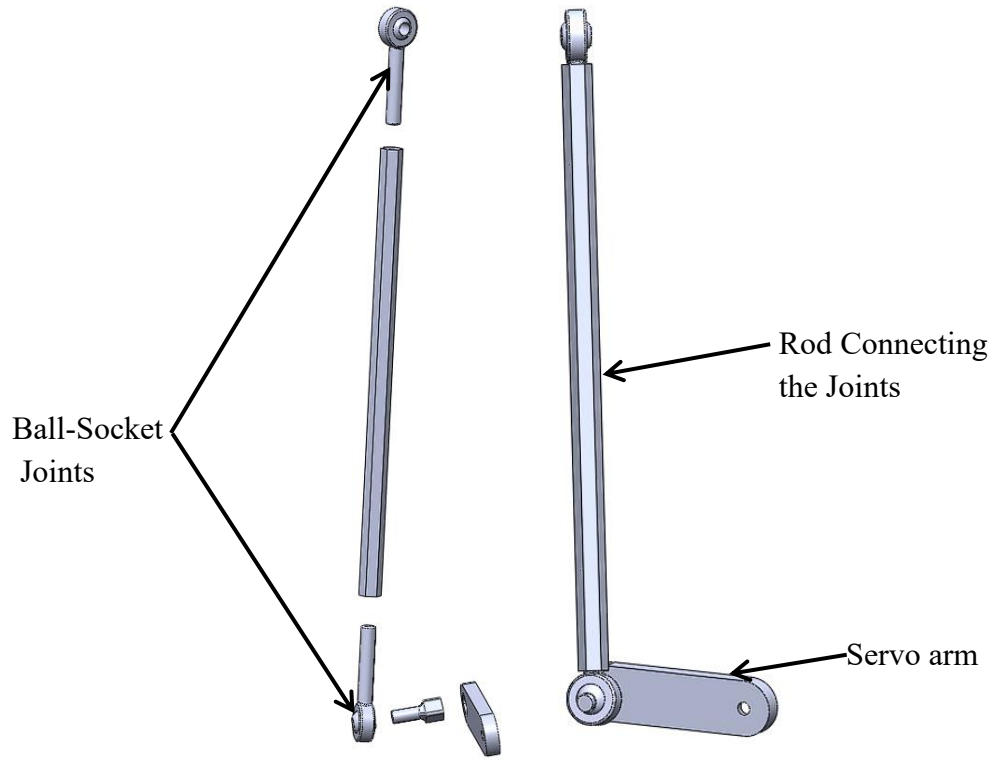


Fig. 33. Components of the linkage subassembly and its exploded view

The final subsection is the mobile platform. This consists of the platform itself, along with six mounting pins which are inserted into it. The joints attach to these mounting pins. The exploded view can be seen in Fig. 34. The feedback sensor or Inertial Measurement Unit (IMU) is also mounted on the underside of the platform.

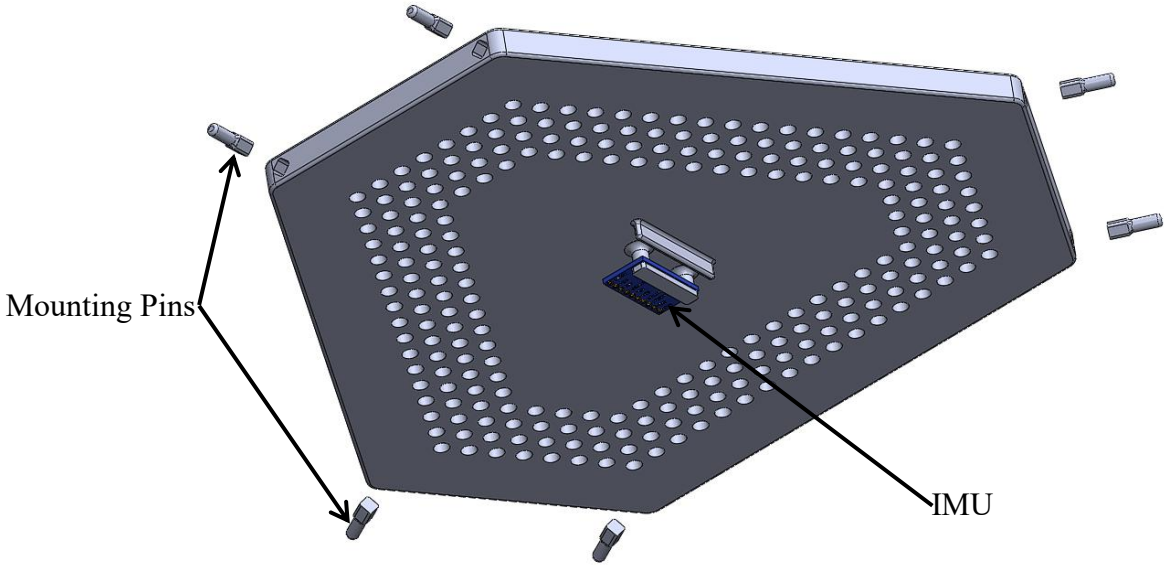


Fig. 34. Underside of the platform showing the mounting pins and the IMU

These are all the different components of this prototype. All these components need to be 3D-printed, except the ball-socket joints.

2.5. Final Prototype Hardware Development

After completing the initial design and visualisation of the final prototype, all the components were 3D printed. Then, the inverse kinematics algorithm was implemented using Python. Finally, the electronics hardware was integrated into the prototype.

The chosen additive manufacturing method for developing these components was FDM (Fused Deposition Modelling), which is the most common additive manufacturing method due to its cost-effective nature and ease of use. The process of FDM builds objects by extruding molten thermoplastic layer-by-layer. It is widely used for rapid prototyping, functional parts, jugs, etc, due to its ease of use and versatility with most common materials such as PLA, ABS and PETG.

All components were 3D-printed using generic PLA (polylactic acid). This is the most popular, user-friendly, cost-effective, and biodegradable thermoplastic filament used for FDM 3D printing. Developed from renewable sources such as corn starch, it is known for its high rigidity and low warping while printing, making it ideal for visual prototypes, hobbyist models and most importantly, for educational projects. PLA has a melting point which ranges from 170°C to 200°C, which is fairly low compared to the other materials. It also has lower stringing and low thermal expansion. These properties reduce the thermal warping of the printed parts, making them more dimensionally accurate. The complete 3D and assembled prototype is as seen in Fig. 35. A very minimalistic colour combination of black and white was chosen for this prototype for aesthetic purposes.

Fig. 35. Developed Final Prototype

As seen above, all the parts have been printed and assembled. The electronics hardware has also been integrated with the printed parts. The next step in the prototype development is the inverse kinematics algorithm implementation. Before getting started with the control algorithm development, it is important to decide the operational range of the platform. After running some motion simulations on the CAD model using the mate controller tool in SolidWorks, the safe operating ranges for all six axes of the platform were identified. Table 3 shows the rotational range, and Table 4 shows the translational range of the platform. The zero position of the platform is when the magnitude of all six axes is equal to zero, or when all the servo arm angles are zero (all servo arms are horizontal to the ground).

Table 3. Rotational range of the platform

Rotational axis	Min, degrees	Max, degrees
Pitch	-10	10
Roll	-10	10
Yaw	-20	20

Table 4. Translational range of the platform

Translational axis	Min, mm	Max, mm
X	-40	40
Y	-40	40
Z	-20	20

2.6. Hardware System Architecture

The second aspect of the prototype is the electronics and hardware integration. This is what gives the Stewart platform the motion capabilities. This section discusses the electronics framework through a

block diagram. The electronics of any mechatronic system determine its response and efficiency. Fig. 36. shows the block diagram for this prototype.

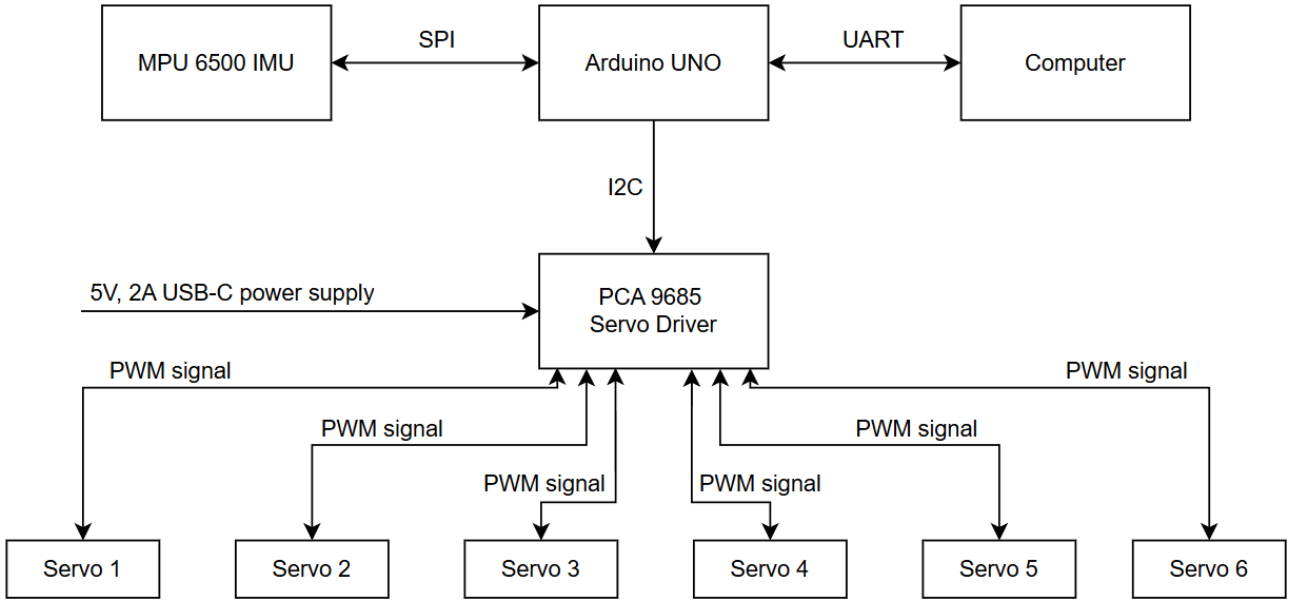


Fig. 36. Block diagram for the final prototype

As seen in the above block diagram, the Arduino UNO is the main microcontroller that handles all servo motor movement operations and processes the feedback data from the IMU. The six calculated servo motor positions are sent to the UNO from the computer as an array of six values via the UART communication protocol. After this, the UNO parses this array of data and converts it into the appropriate movement angles and then sends this data to the PCA 9685 Servo Driver through the I2C communication protocol. This driver then converts these six angles into individual PWM signals and sends these signals to the respective servo motors. Once these servo motors move to the desired position, the current positional data is extracted from the feedback IMU sensor and any error corrections in the position of the platform are performed. At this stage, one loop of the entire platform is complete. This loop keeps on going until the position arrays keep coming from the computer. Fig. 37. shows the flowchart of the sequence of tasks that the hardware undertakes.

This flowchart helps with understanding the flow of tasks in the hardware of the platform. The first task after powering on is to establish all the communication protocols. The computer communicates with the UNO through the UART protocol. This is a standard method for interfacing computers with most microcontrollers. The UNO then establishes I2C communication with the servo driver. Here, the desired positions of the six servo motors are transmitted to the drive, after which it moves each of them to its respective position. The IMU is connected to the microcontroller through the SPI protocol. Then the UNO checks whether the UART interface is terminated. After this, the UNO checks if it received any input arrays from the computer. If yes, the input is processed, the servo angles are calculated, and finally, they are moved. After they are moved to their desired position, a check is performed to detect any positional errors. If yes, they are corrected for using a PID control. After this, the loop repeats, and the UNO waits for the next input array to be transmitted.

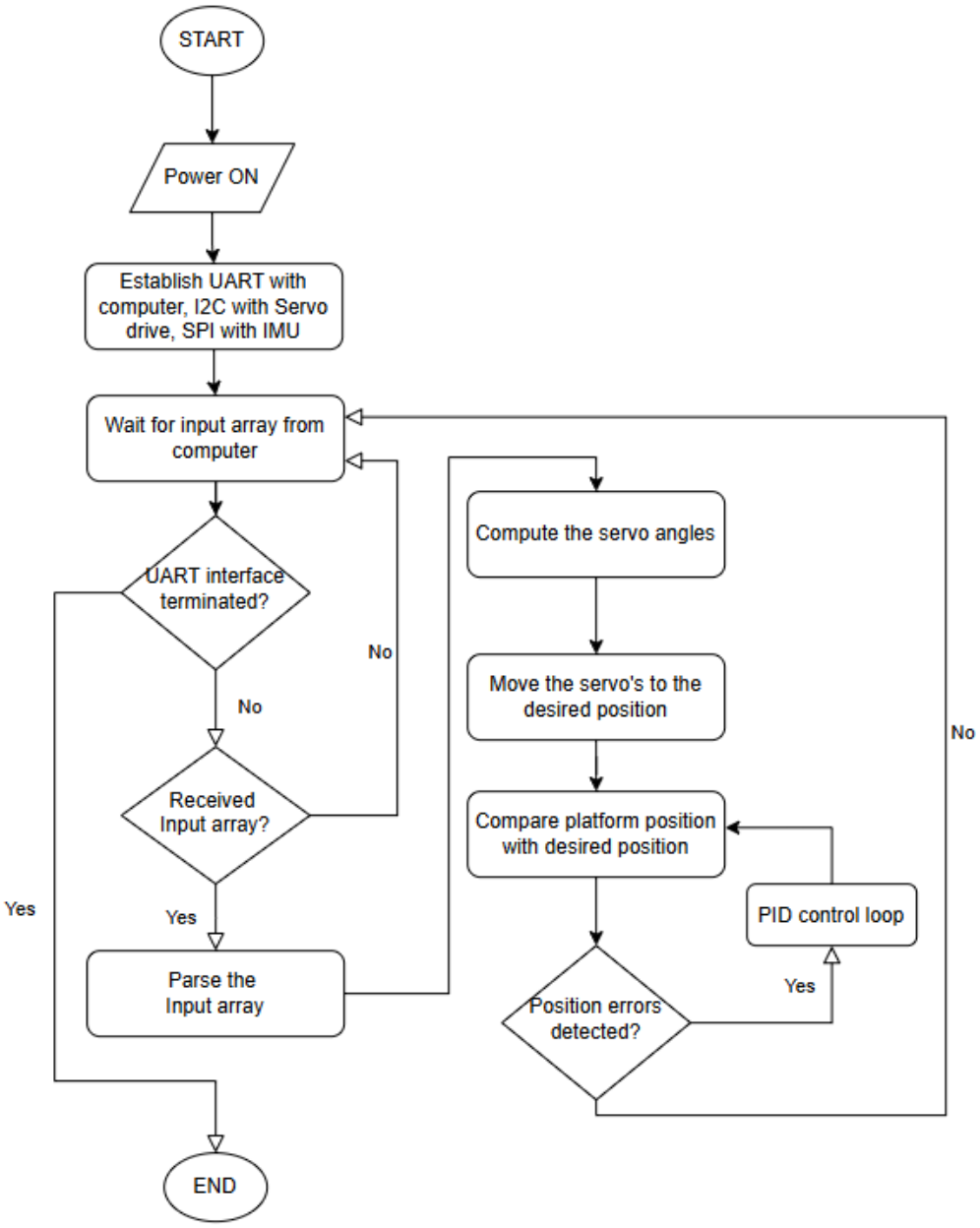


Fig. 37. Programme flowchart

2.6.1. Servo Motor Selection

This prototype uses rotary servo motors to move the platform. These motors have to be relatively precise, and at the same time, they should not be too expensive. Hence, the MG995 servo motor was chosen for this project. This is a hobbyist-level motor which offers good reliability and repeatability. This motor also has a good payload capacity and requires very minimal setup time. It has a maximum operation voltage of 7.2V and works using PWM signals. Servo motors are controlled using PWM signals. PWM works by sending a digital signal to the signal wire of the motor, where the width of the pulse (0.5ms-2.5ms) determines the angular position of the servo motor shaft. Fig. 38. shows the MG995 servo motor used, along with its dimensions and the wires. A total of six motors were used in this prototype. Table 5 shows the mechanical and electronic properties of the servo motor.

Table 5. Mechanical and electrical properties of the servo motor

Property	Value
Operating speed	0.13 s/60degrees without load
Operating voltage (DC)	3.0V – 7.2V
Operating current	350mA
Stall torque	13 kg/cm
Stall current	1500mA

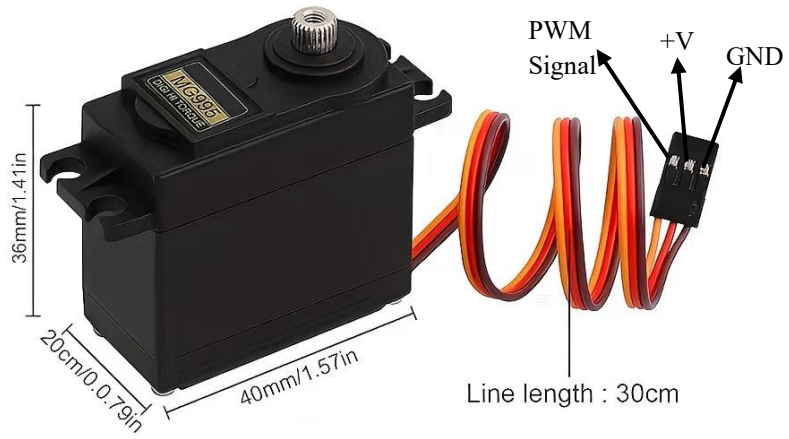


Fig. 38. MG996 Servo motor

Although the servo motor comes with its stock attachments, they were not very useful for this application. Hence, a custom servo arm was designed. This arm attaches between the lower ball-socket joint and the servo shaft. Its design is shown in Fig. 39. This arm was structurally tested to ensure it can withstand the shear and bending forces induced while moving the platform. Considering that the moving platform weighs 200g.

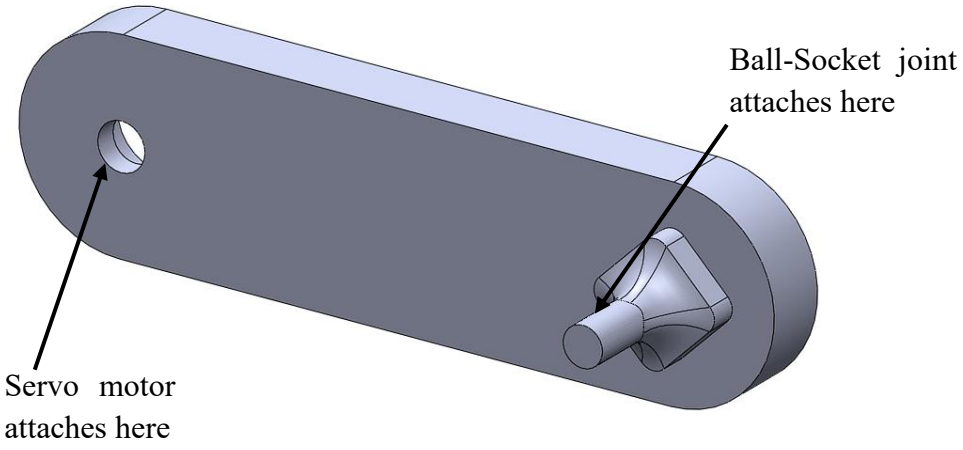


Fig. 39. Servo arm design

Since SolidWorks does not have a default PLA material in its library, a custom material with the mechanical properties of PLA was created. The properties can be seen in Table 6.

Table 6. Mechanical properties of PLA material

Parameter	Value	Unit
Elastic Modulus (Young‘s)	3500	N/mm^2 (MPa)
Poissons Ratio	0.35	NA
Mass Density	1245	Kg/m^3
Tensile Strength	50	N/mm^2 (MPa)
Yield Strength	38	N/mm^2 (MPa)
Thermal Expansion	4.1e-05	/K

The structural analysis was carried out on SolidWorks. The total weight of the moving platform is 200g. Assuming a safety factor of 1.5, a load of 300g was applied on the ball-socked joint side, and the other side was fixed. This can be seen in Fig. 40. along with the generated mesh.

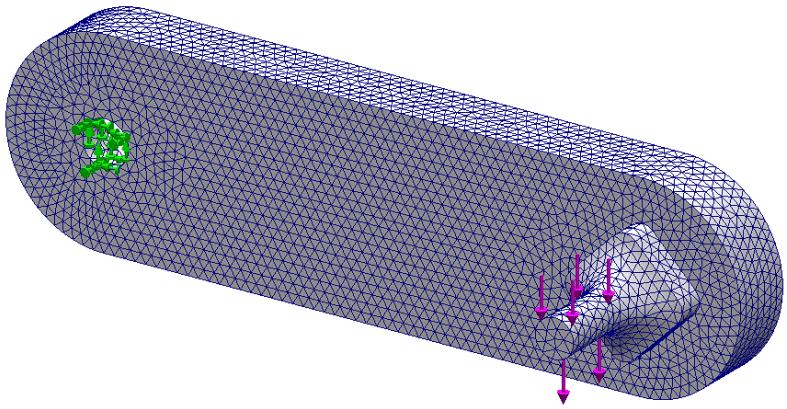


Fig. 40. SolidWorks simulation setup

The simulation was run, and the following stress and displacement plots are seen in Fig. 41.

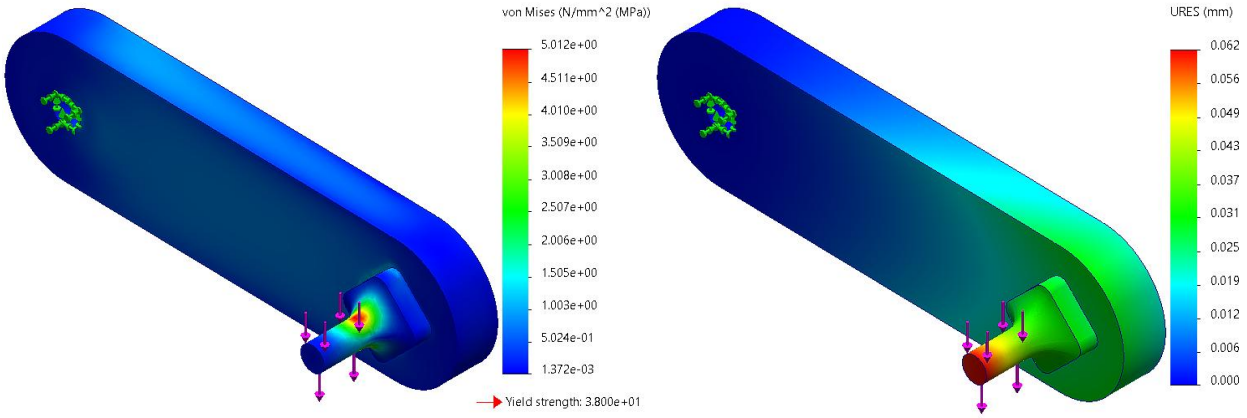


Fig. 41. a) Resultant stress plot, b) Resultant displacement plot

As observed from the above result plots, it is observed that the maximum stress experienced is 5.012MPa, which is much lower than the yield strength of 38MPa. It is also observed that the resulting displacement is 0.062mm. This is a considerably small magnitude of displacement. Hence, this proves that the designed servo arm is structurally strong to handle the forces which will arise during the operation of the platform.

2.6.2. Servo Motor Driver

As seen in Table 5, each servo motor needs a constant voltage supply between 3 and 7.2V. This becomes a problem if 6 of these servomotors have to be controlled at a time while supplying the appropriate power to them. Hence, a dedicated motor driver which uses the PCA9685 chip was used. This is an electronic module which can control up to 16 servo motors at a time. Fig. 42. shows the servo driver module. This board has a supply voltage of 5 to 10V, which is perfect for the chosen servo motors. This module communicates with the microcontroller through the I2C communication protocol. The Arduino UNO sends the calculated servo angles to this module, after which it ensures that each servo has reached its desired position.

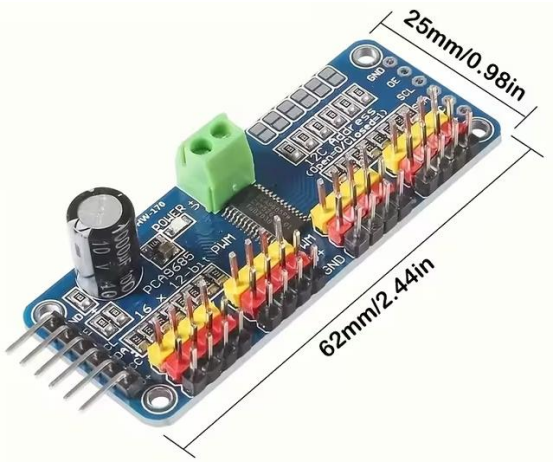


Fig. 42. PCA9685 Servo motor driver module

2.6.3. Microcontroller System

As mentioned in the above sections, the chosen microcontroller for the final prototype of the Stewart platform is an Arduino UNO, specifically the R3 variant. This is an open-source microcontroller board which acts as the 'brain' for the platform. Arduino UNO is considered the standard when it comes to education because it was designed specifically for students and non-technical creators. Fig. 43. shows the microcontroller used along with all its General Purpose Input Output (GPIO) pins.

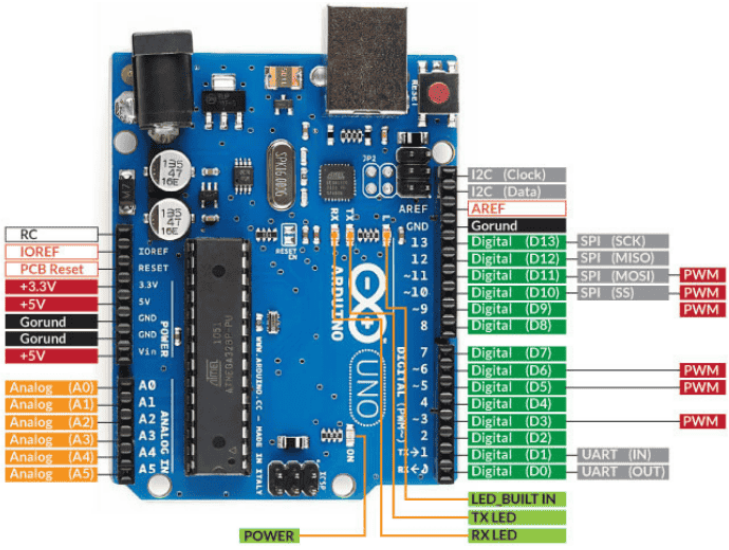


Fig. 43. Arduino UNO R3

Known for its plug-and-play nature, UNO connects directly to a computer via a standard USB cable that handles power and data. It uses a simplified version of C++, making it an ideal first programming language for students who have never coded before. The UNO is also built with remarkable robustness and can survive some common beginner mistakes, such as short-circuits that would destroy other sensitive boards. This board is also quite inexpensive compared to its competitors due to its open-source nature, making it ideal for educators all around the world. This openness has also created a massive global community, which allows for unlimited access to free, ready-made lesson plans and project plans. Another advantage is the abundance of libraries, which are pre-written code blocks which allow students to control complex electronic components like screens, sensors and motor drivers without having to figure out each line of code on their own, which could discourage many students.

2.6.4. Feedback Sensor Selection

The final electronic aspect of the prototype is the feedback sensor. Positional feedback is essential for a Stewart Platform to transform it from a passive machine to a precise closed-loop mechatronic system. Because Stewart platforms are used in high-precision applications such as flight simulators and aural equipment, they require active control to compensate for mechanical aberrations and real-world disturbances. When it comes to positional feedback, mainly six values are needed. They are the deviations from the desired position of the platform in the form of the six axes.

An Inertial Measurement Unit (IMU) is used in applications which require positional feedback. This is an electronic module which measures a body‘s specific force, angular rate and sometimes the magnetic field surrounding it. It is essentially the 'inner ear' of a closed-loop mechatronic system, telling the controller how it is moving and how it is oriented in 3D space. The IMU typically combines three specific sensors into one package. First is the accelerometer, which measures linear acceleration to compute the tilt of an object (pitch and roll). Next is a gyroscope that measures angular velocity to track a change in rotation. Third is a magnetometer, which acts as a compass to determine heading or Yaw of an object relative to the Earth's magnetic field. The chosen module for this project was the MPU-9250/6500 board, as seen in Fig. 44. This board communicates with the microcontroller through the SPI communication protocol.

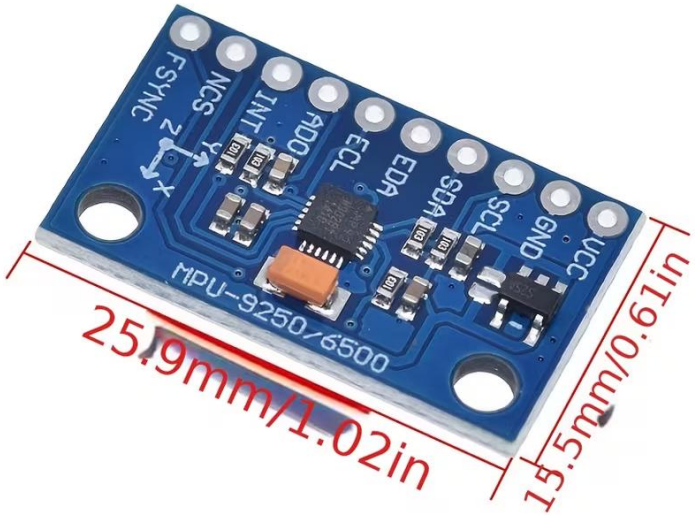


Fig. 44. Chosen IMU module (MPU-9250/6500)

2.6.5. Component Layout and Power Supply

All the components except the IMU are placed in the base housing unit. These include the six servo motors, Arduino UNO and the PCA9685 servo motor driver. The layout is as seen in Fig. 45.

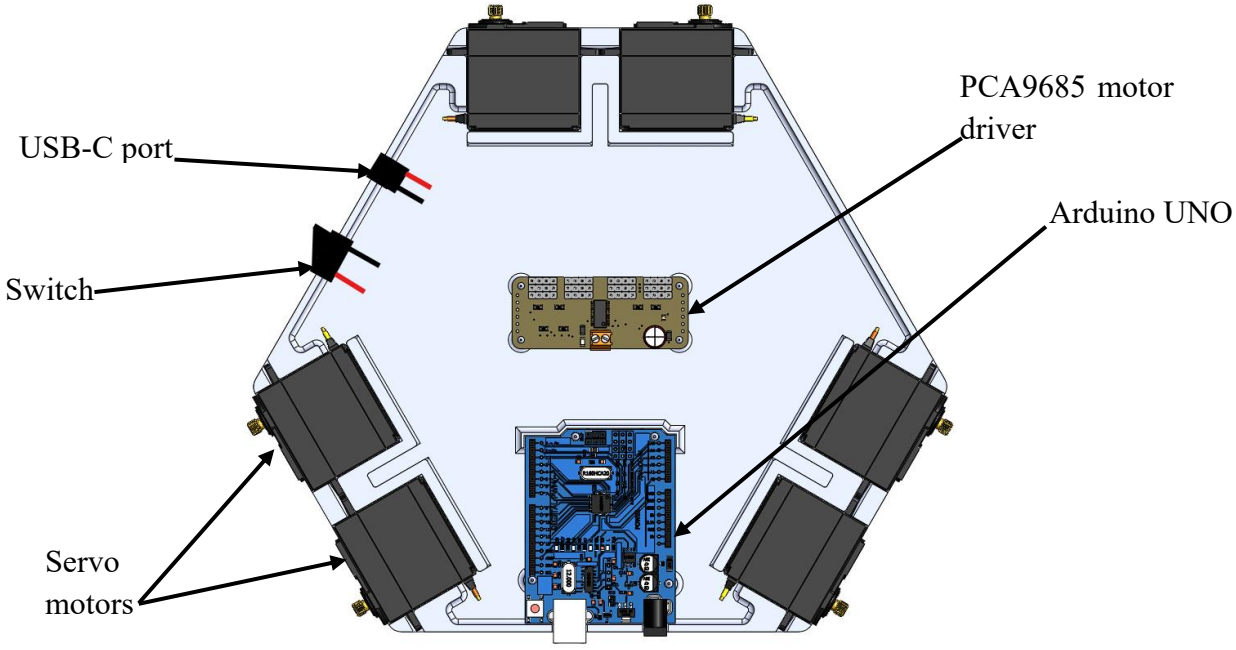


Fig. 45. Electronic components layout

Coming to the power supply, this prototype requires most of the power for the six servo motors. The PCA 9685 module provides the necessary power to these motors. As seen in the above figure, the USB-C port is used to supply a constant DC voltage to the motor driver. The next step is to determine the total operational power consumption of the system. This mainly includes the servo motors and the Arduino UNO. The equation for calculating the power required for all six servo motors is as seen in Equation 15.

$$P = V * 6(I_m), \tag{15}$$

Where: P – total power consumption of the six motors
V – voltage being supplied to the motors
I_m – current consumption of a single motor

Considering that each motor has a load of 40g, and the motor has a stall torque of 13kg/cm, it is safe to assume that each motor will not exceed the operating current rating of 1A when supplied with a voltage of 6V. Therefore, the total power consumption becomes.

$$P = 6 * 6(1), \tag{16}$$

$$P = 36W. \tag{17}$$

The total calculated power required for this system is 36W. Which means a generic phone charger with a power output of more than 36W, along with an output voltage of 6V, will be able to provide this power.

2.7. Inverse Kinematics Algorithm Implementation (GUI)

After assembling the prototype, the next step was to control it. Hence, a Graphical User Interface (GUI) was created using Python. A GUI is a type of user interface that allows users to interact with electronic devices through icons, buttons, sliders and menus rather than having to write code every time. In this project, the GUI acts as a 'dashboard' on the computer, allowing the students to visualise the platform, adjust its parameters and control it in real-time without needing to reupload code. The developed GUI is as seen in Fig. 46. The main requirements of the developed GUI are:

a. Individual axis controls: The main requirement of the GUI was to be able to move the platform in each of the six axes individually. This will allow for an understanding of how the platform functions and what its workspace is.
b. Digital twin: A digital replica of the platform needed to be made using the above-mentioned mathematical model. This digital twin will aid in comparing the movements of the platform with the calculated values. This can be achieved by plotting the top and bottom coordinates of each linkage and connecting those points together.
c. Dynamic movement capabilities: There was a requirement for inducing dynamic motions in the platform, such as oscillations in one or more axes. Here, the dynamical parameters such as amplitude, frequency and phase of a particular oscillation can be studied and understood by the students.
d. Real-time servo motor position display: There was also a requirement to see the computed desired position of the servo motors in real time as the platform moves.
e. Ease of use: The GUI needed to be easy to use and straightforward to understand.

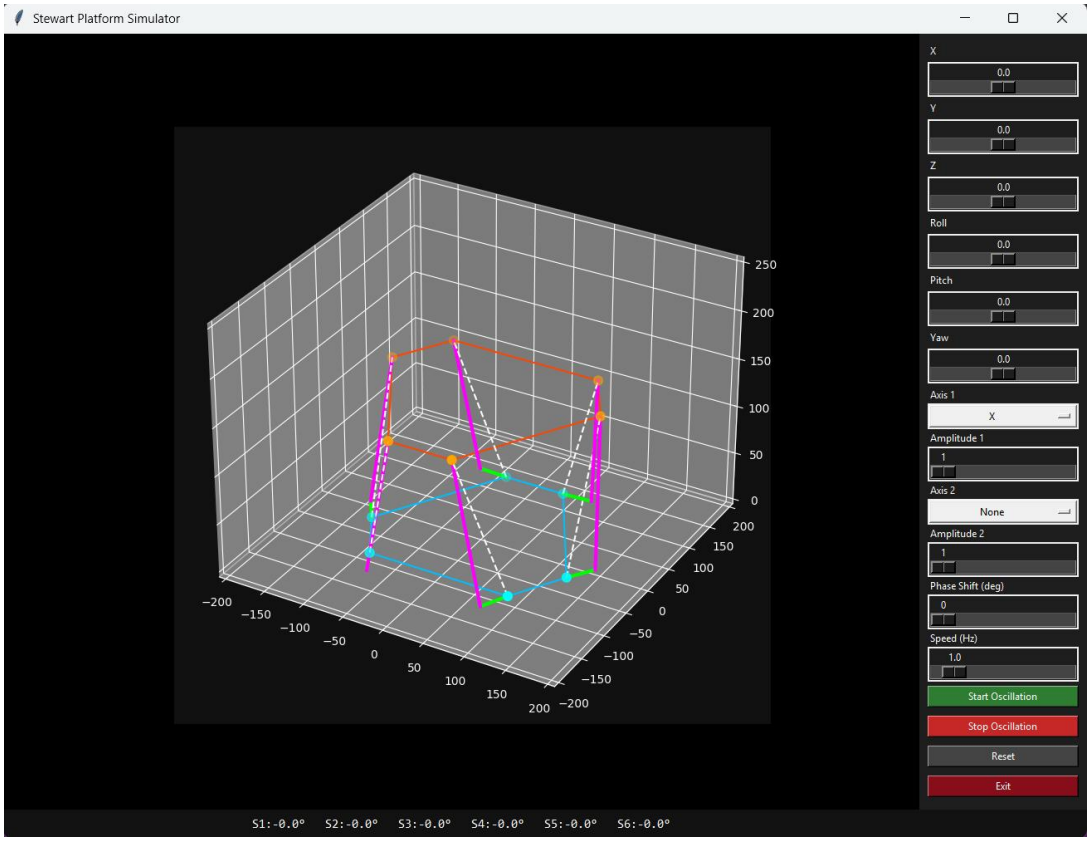


Fig. 46. The developed GUI for the Stewart Platform

This GUI consists of a 3D graph in the centre where the coordinates of the platform are plotted. The orange points on the top are the points where the upper ball-socket joint of a linkage attaches to the moving platform, and the light blue points on the bottom are the points where the servo shafts attach to the servo arms. The orange lines form an outline frame of the top moving platform, and the blue lines form an outline frame of the base of the platform. All the green lines represent the servo arms, and all the pink lines represent the rod which connects the two ball-socket joints of each linkage.

On the right side of the GUI, there are the different controls used to manipulate the platform. The first six sliders are used to move the platform in their respective axes (X, Y, Z, Roll, Pitch, Yaw). Below them is the dynamic motion section. Here, there are dropdown menus along with their amplitude sliders, allows for the selection of any two axes along with their amplitudes. Then there are the phase shift and speed sliders. This section enables the oscillation of the platform in any two axes with the desired amplitude, speed and phase shift between them. Below them, there are the start and stop buttons for the oscillation. Then the reset button, which simply resets all the axes sliders to 0, making the platform return to its home position (when all the servo angles are 0). Finally, an exit button that closes the GUI and exits the loop.

As mentioned above, the GUI was programmed in Python. This Python code can be broken down into two main sections or classes. The first class handles the inverse kinematics calculations for the platform and transmits the calculated servo angles to the Arduino UNO through the USB port using UART communication. The second class is the GUI itself, which is the front-end part of the code. The working principle is that the servo angles keep getting calculated every time there is a change in the input values of the six-axis sliders in the GUI. Once it is calculated, the values are sent to the UNO. Fig. 47. shows a flowchart for the Python code.

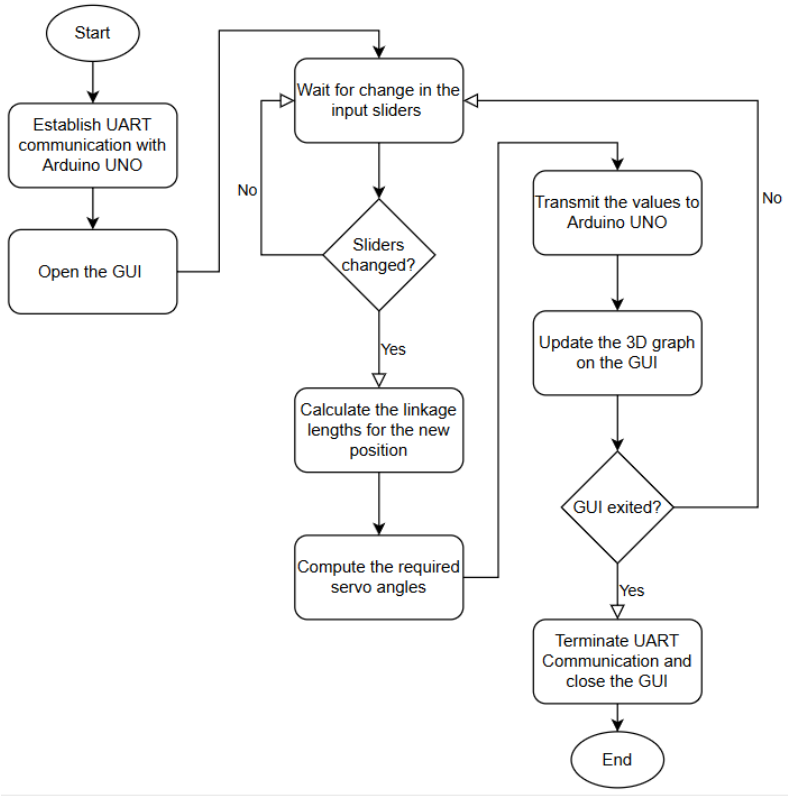


Fig. 47. Flowchart for the Python code and GUIs

The servo angles are calculated using the inverse kinematic model described in the sections above. The coordinates of each joint on the fixed base and moving platform are indicated in the top view of the platform in Fig. 48. Light blue points are on the fixed base, and orange points are on the moving platform. The black point in the centre is the origin, which is located in the base plane.

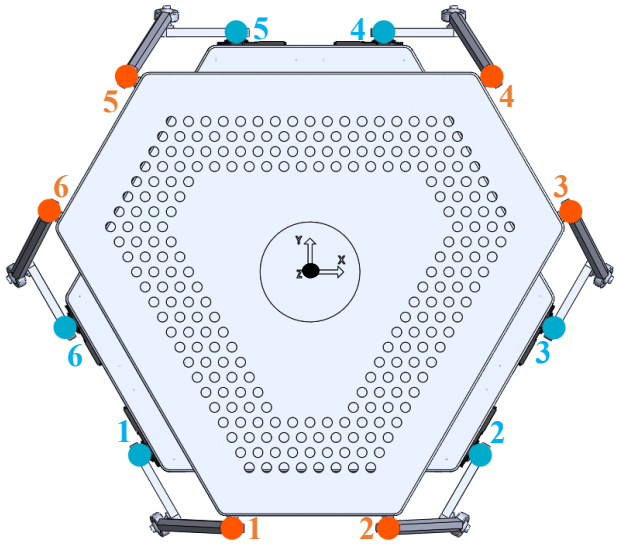


Fig. 48. Base and platform joint coordinates numbering

The values of these coordinates are indicated in Table 7 and Table 8.

Table 7. Fixed base coordinates used (Light blue points)

Joint Number	x, mm	y, mm	z, mm
1	-87.11	-93.02	0
2	87.11	-93.02	0
3	124.11	-28.93	0
4	37	121.95	0
5	-37	121.95	0
6	-124.11	-28.93	0

Table 8. Moving platform coordinates (Orange points)

Joint Number	x, mm	y, mm	z, mm
1	-39	-128.19	145
2	39	-128.19	145
3	130.52	30.32	145
4	91.52	97.87	145
5	-91.52	97.87	145
6	-130.52	30.32	145

With the help of this GUI, the students will be able to learn how a Stewart platform functions by physically moving it around. They can also understand dynamics, simple harmonic motions, and oscillations through the dynamic controls of the platform. The next section discusses the educational part of this prototype in more detail.

3. Educational Laboratory Framework

The developed prototype of the Stewart platform is intended for educating the concepts of mechatronics engineering to bachelor's level students. Since this is not a very simple machine, this lab work is better suited for students in their third and fourth years, once they have developed some programming skills and have a better understanding of mathematical fundamentals, such as matrix manipulation and vector algebra. This educational platform aims to teach and reinforce the concepts of mechatronics engineering, such as mechanical design, mathematical modelling, electronics and control systems engineering. These concepts are an integral part of mechatronics engineering, and having experience working with a Stewart platform can be highly useful for a student after graduating.

Before getting started with the exercises, it is important to establish the prerequisites which the students need to have. The students must have a basic understanding of CAD modelling and assemblies. Some mathematical prerequisites, such as vector algebra, linear algebra, rotation matrices, transformation matrices and basic trigonometry, are needed. The students also need to have an understanding of kinematic concepts such as coordinate systems, rigid-body transformation and 6-DOF understanding. Lastly, the students need to have computational skills, such as geometry definition, where they would need to know how to locate the six base attachment points and the six platform attachment points and programming skills, which involve the implementation of the mathematical model in software such as MATLAB. The proposed lab work is split into three exercises. Each focuses on teaching one aspect of mechatronics engineering.

3.1. Mechanical Structure and Assembly of The Stewart Platform (Exercise 1)

This exercise aims to develop a foundational understanding of the mechanical design and assembly of a Stewart Platform and how its physical configuration enables six degrees of freedom (6-DOF) motion. Here, the students learn what a Stewart Platform actually is, along with its applications and the different versions of it available. The students will also learn how the geometry, servo motor placement and joint arrangement influence the platform‘s motion, rigidity and workspace.

3.1.1. Theoretical Background

Here, the students learn about the different fundamental components of a Stewart platform, such as the fixed base, movable platform and the six independently actuated linkages which connect the base and the platform. This is done through the CAD model of the platform, as seen in **Fig. 31**. Each student has to go through the CAD model and understand the function of each component. They can also use tools, such as the Mate controller, to move the platform and get an understanding of how the servo motors influence the movement of the platform. The six degrees of freedom are also explained here. Stating that it consists of three translational (linear) motions along the x, y, and z axes and three rotational motions (Roll, Pitch and Yaw).

3.1.2. Mechanical Concepts

Here, the students will receive the platform partially disassembled. They must identify and examine all the components. Table 9 indicates each component to be analysed and the key observations to be made when examining them.

Table 9. All the components to be analysed, along with the key observations to be made

Component name	Description	Key observations
Fixed Base	This is the stationary lower structure supporting the system	Shaper and rigidity, Mounting points, Symmetry, Linkage attachment locations
Moving Platform	The upper platform responsible for motion generation	Platform geometry, Joint placement
Linkages	These generate motion by changing their lengths	Linkage structure, Lengths of the servo arm and rod connecting the joints
Joints	The joints allow multi-axis movement between the links, base and platform	DOF allowed, Mechanical play, Joint limits
Servo motors	The Servo motors change the length of the linkages by changing their angles.	Linkage mounting points, Numbering of the motors and placement in the base

Next, the students should measure and record some dimensions and coordinates which will be used in the later exercises. These include the base radius, top platform radius, neutral platform height, linkage length limits, and attachment point coordinates. All these dimensions and coordinates can be easily measured and recorded using the CAD model on SolidWorks. The coordinates are as seen in Table 7 and Table 8.

3.1.3. Assembly procedure

Once all the components are inspected, along with the key observations and measurements, the assembly procedure can begin. All the components are assembled with the help of a step-by-step table of instructions, as seen in Table 10 and also with the help of the CAD model.

Table 10. List of instructions for assembling the Stewart Platform

Step No.	Step name	Description
1	Initial inspection	An initial check is performed to ensure that all the components are present as listed in the provided part list.
2	Base sub-assembly	All the electronic components, such as the servo motors, servo drive and the Arduino UNO, are assembled in the base unit as seen in **Fig. 45**.
3	Linkage sub-assembly	All the linkages are assembled using the ball-socket joints, servo arms and the connecting rod as seen in **Fig. 33**.
4	Top platform sub-assembly	All the mounting pins are inserted into the top platform, and the IMU is mounted on it too, as seen in **Fig. 34**.
5	Final assembly	Finally, all these subassemblies are brought together and assembled. The CAD model can be used as a reference while making the final assembly.

3.1.4. Expected Learning Outcomes

At the end of this exercise, the student should understand what a Stewart Platform is, its architecture and should also be able to explain 6DOF mechanical motion. They should correctly assemble the entire platform, recognise geometric constraints and finally, should be able to relate platform geometry to motion behaviour.

3.2. Electronics and Wiring (Exercise 2)

This exercise teaches the electronics architecture of the Stewart Platform and teaches students how electrical systems interact with mechanical systems to produce controlled motion. Here, the students must have a basic understanding of electronics and the different communication protocols used, such as UART and I2C. Students will get to learn the concepts of power distribution, signal generation, controller interfacing and sensor integration. A very important concept needed to be understood before getting started with this exercise is how Pulse Width Modulation (PWM) signals work. As the servo motors are controlled through PWM signals

3.2.1. Electronic system overview

This Stewart Platform consists of a microcontroller, servo motors and a servo motor driver to control them. An Inertial Measurement Unit (IMU) is used for the positional feedback of the platform. This will be learned in exercise 4. The block diagram for the electronic system is shown in **Fig. 36**. Students should understand digital outputs, PWM signals, timing control, and Serial communication before getting started with the programming of the Arduino UNO.

3.2.2. Wiring Procedure

After understanding the electronic architecture and the block diagram, the students can start wiring up the system. Table 11 shows a step-by-step procedure for the wiring needed to be done. Fig. 49. shows the circuit diagram which can be used while wiring.

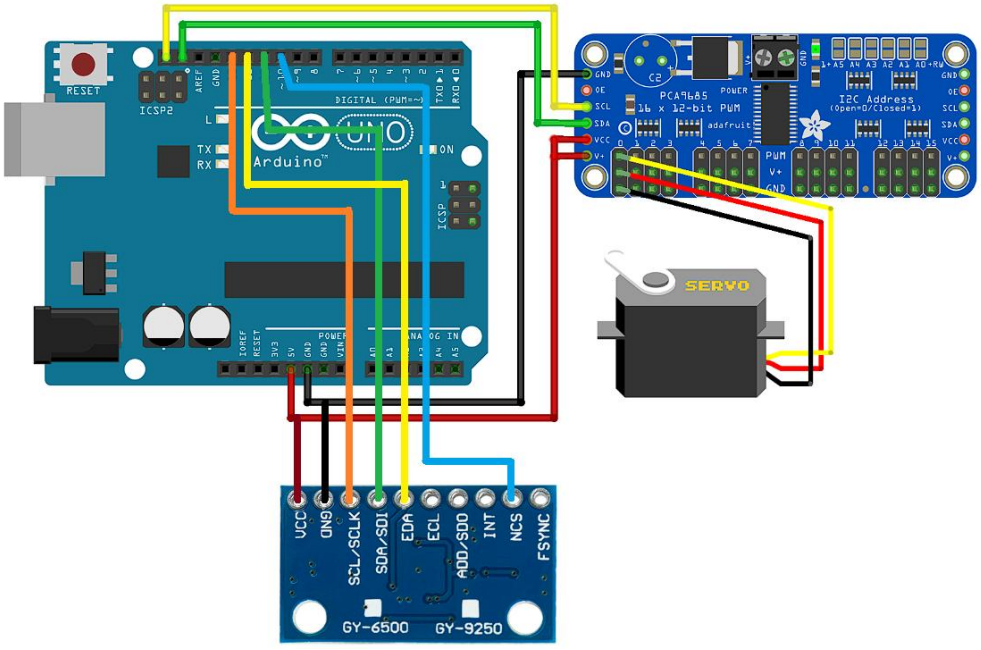


Fig. 49. Circuit diagram showing the wiring of the electronic components

Table 11. List of instructions for wiring the electronic components

Step No.	Step name	Description
1	Servo motor wiring	All the servo motors should be connected to the motor driver in a sequential order.
2	Servo driver wiring	The servo driver should be connected to the Arduino UNO pins as seen in the circuit diagram.
3	IMU wiring	The IMU board is connected to the Arduino UNO and uses the SPI communication protocol.
4	Power connections	Connect the USB-C port with the switch. Verify voltage polarity and establish a common ground.
5	Progam uploading	Upload the provided C code onto the Arduino. Code in Appendix 1

3.2.3. Expected Learning Outcomes

By the end of this exercise, students should understand Stewart Platform electronics. Safely wire the whole system, generate PWM signals and control the servo motors, learn to diagnose any electrical issues if they arise and upload the provided C code onto the Arduino UNO.

3.3. Mathematical Modelling and Inverse Kinematics (Exercise 3)

This exercise aims to develop the mathematical framework necessary to describe the Stewart Platform motion. Students will learn how the desired platform motion and position are converted into linkage lengths and, in turn, the required servo angles using inverse kinematics. The students will be taught the formation of the mathematical model in a step-by-step manner, after which their task will be to try to replicate the model through code on any programming platform, most preferably MATLAB.

3.3.1. Coordinate Systems, Rotation Representation and Inverse Kinematics

The first step in the mathematical model is to define the base coordinate frame and the moving platform coordinate frame. These can be done by plotting the recorded coordinates in a 3D plot from exercise 1. Then, the students must define the base attachment points B_k and the top platform attachment points P_k. These coordinates determine the platform geometry. Then, students should study Euler angles, rotation matrices, and coordinate transformations and try to replicate the mathematical model defined from Equation 1 through Equation 14. Inverse Kinematics is the process of calculating the necessary servo motor angles for the desired platform position. This makes the servo angles the output of the mathematical model, and the desired position of the platform becomes the input. The students can make use of platforms like MATLAB, Python and Simulink to create the mathematical model. This is one of the most important parts of the laboratory because students learn how mathematics becomes executable control logic.

3.3.2. Expected learning outcomes

By the end of exercise 3, students should understand how a rigid body is represented mathematically, why coordinate transformations are essential, how geometry defines controllability, learn to implement mathematical models computationally and why Stewart Platforms require continuous inverse kinematics computation.

3.4. Platform Control and Motion Analysis (Exercise 4)

This exercise integrates all previous exercises into a complete Stewart Platform operation. Here, students get to use a prewritten control software (Source code provided in Appendix 2) to execute platform motion, observe dynamic response and analyse system performance. The main objective here is to learn how the software computes the servo angles every time there is a change in the input parameters of the platform. This exercise emphasises experimental engineering, measurement and system analysis.

3.4.1. Motion Generation

A Stewart Platform can generate a wide variety of motions, but they can be mainly categorised into three types. They are categorised based on the type of motion being generated, as seen in Table 12.

Table 12. Different types of motions on a Stewart Platform

No.	Motion name	Description
1	Translational motion	This is the linear movement of the platform along the three Cartesian axes. Forward/back (X), left/right (Y) and up/down (Z).
2	Rotational motion	This is the rotating motion along the three Cartesian axes. Tilt front/back (Pitch), tilt left/right (Roll), Twist left/right (Yaw).
3	Combined motion	This motion is a combination of the first two and the most realistic movements which can be expected out of a Stewart Platform. Eg: moving upward while pitching forward

The students will need to induce these three different motions into the Stewart Platform and observe the synchronised actuator movements and workspace boundaries. Students can also generate trajectories using the Stewart Platform. This can be done by using the oscillation function in the GUI, as seen in . They can generate sinusoidal or circular oscillations in one or two axes at a time and observe how the platform generates these movements.

3.4.2. Experimental measurements

The primary purpose of this section is to compare the previously calculated theoretical predictions to the actual measured response of the platform under real operating conditions. Students first define a desired pose sequence or trajectory, such as vertical translation, forward tilt or sinusoidal oscillation along any axis or even a combined 6DOF motion and then record how the physical platform behaves over time. Measurements may include the servo angles, platform displacement or angular orientation data, which can be obtained using the encoders on each servo motor and the IMU installed on the underside of the moving platform. The data from these sensors can be extracted and plotted alongside the calculated values to compare the difference between them.

3.4.3. Expected learning outcome

By the end of exercise 4, students should have developed a comprehensive understanding of how a Stewart Platform operates as an integrated mechatronics system that combines mechanics, electronics, mathematics and control engineering. They should be able to operate a Stewart Platform safely, understand its full control architecture, interpret servo motor behaviour, diagnose practical limitations and most importantly, connect theory with experimental reality.

4. Social and Economic Aspects of the Project

A low-cost Stewart Platform serves as a high-impact mechatronics teaching tool by providing access to multiple teaching disciplines. It allows students to progress from abstract theory and computer simulations and engage in real-world robotics, kinematics and control theory without the high costs of industrial teaching tools, which often cost tens or hundreds of thousands of euros. Low-cost platforms enable smaller universities and those with limited budgets to provide advanced robotics and mechatronics labs, thereby levelling the playing field for students studying there. A low-cost Stewart Platform built from 3D-printed parts, hobby servos, low-cost sensors, and cheap microcontrollers like Arduino can reduce costs to 100-200 EUR. This opens the doors for advanced mechatronics education in public schools, vocational institutions and universities in low-income regions. A major contributor to the low cost of this project is the fact that all the designed mechanical parts have been 3D printed. The chosen method of 3D printing was Fused Deposition Modelling (FDM) due to its simplicity and affordability compared to the other 3D printing methods available in the market now. The chosen material for this project was PLA. The average cost of filament-based 3D printing material like PLA is anywhere between 0.02 and 0.10 EUR per gram. Table 13 shows a list of all the 3D printed parts along with their quantity, weight and cost.

Table 13. Cost of all the 3D printed parts

Part name	Quantity	Weight, g	Cost, EUR
Base	1	218.88	4.38
Base Cap	1	129.89	2.6
Servo Arm	6	10.39	0.21
Mounting Pin	12	3.12	0.06
Top Platform	1	173.38	3.47
Connecting Rod	6	18.27	0.37
Stand	3	30.61	0.61
Axis Indicator	1	2.27	0.05
Total	-	**586.81**	**11.75**

As seen from the table above, the total cost of all three 3D printed parts is just under 12EUR. The next contributor to the cost of the project is all the electronic components used. Table 14 shows a list of all the electronic components used, along with their quantity and cost.

Table 14. Cost of all the electronic components used

Component name	Quantity	Cost, EUR
Arduino UNO R3	1	28.60
PCA 9685 Servo Driver Module	1	4.18
MG995 Servo Motor	6	50.4
MPU-9250 9-DOF IMU Sensor Module	1	2.30
Ball and Socket Joint	12	3.05
Wiring and Soldering	-	5
Total	-	**127.08**

The total cost of the entire hardware, including the 3D printing costs and electronic components, comes up to 140EUR.

Conclusions

1. A thorough investigation of existing mechatronics teaching platforms was carried out, where the main focus was on identifying educational platforms that incorporate physical systems to provide a more hands-on and experiential learning experience for the students. A few existing systems, such as PlatROB and LEGO Mindstorms, were found. Additionally, the use of Stewart Platforms in engineering education was investigated, and it was revealed that most of these systems are limited to theoretical teaching and computer simulations due to their high cost and complexity. Therefore, the necessity for a novel, low-cost and simple Stewart Platform intended for teaching through lab exercises was revealed. The reason for choosing a Stewart Platform over other machines is due to its multidisciplinary nature, as it integrates concepts of mechanics, electronics, computer programming and control systems engineering into a single system.
2. Based on the findings of the literature review, a primary prototype of the Stewart Platform was designed using SolidWorks. This design process primarily focused on the mechanical structure of the platform. Here, some key performance characteristics of this Stewart Platform, such as the range of motion and servo angle limitations, were subsequently analysed. The Mate controller tool on SolidWorks was employed to independently control the angle of each servo motor. This enabled the visualisation and evaluation of the Stewart Platform's kinematic behaviour in response to the different servo angles.
3. Following the design of the primary prototype, the platform geometry was analysed, and a mathematical model was derived. This mathematical model made use of inverse kinematics, which is the process of determining the required linkage lengths for the desired six-degree-of-freedom pose. Since the chosen design of the Stewart platform uses servo motors instead of linear actuators, an extra step of calculating the servo angles was also necessary.
4. Based on the insights gained from the previous tasks, a fully operational prototype of the Stewart Platform was developed. The mechanical components, including the base, base cap, and top platform, were fabricated using the fused deposition modelling (FDM) method, and polylactic acid (PLA) was the chosen printing material. The electronic control system was centred around an Arduino UNO, which receives control inputs from the computer. The UNO controls six servo motors via an intermediate servo driver module. An inertial measurement unit (IMU) was integrated to provide feedback on the platform's position and orientation while operating. Furthermore, a graphical user interface (GUI) developed in Python employed the derived mathematical model to calculate the required servo angles for the desired platform position in real-time.
5. A laboratory programme was proposed that employed the developed Stewart Platform prototype. This programme was divided into four distinct exercises. Exercise 1 focuses on teaching the mechanical design aspects of the Stewart Platform by allowing the students to investigate all the components and assemble the platform with the help of a step-by-step guide. Exercise 2 introduces the electronics architecture of the Stewart Platform, in which students are provided with the block diagram, where they are required to understand the electronics framework of the system and complete the circuit wiring. Exercise 3 addresses the mathematical framework necessary for the motion control of the Stewart Platform by requiring the students to implement the mathematical model by writing code. The fourth and final exercise integrates the previous exercises by providing the students with a prewritten control software to execute platform motion, observe dynamic response, analyse system performance and evaluate overall system performance.

List of references

1. SALES GONÇALVES, Rogério. *Application of LEGO Mindstorms Kits for Teaching Mechatronics Engineering [*online*]*. 2017. Available from: http://www.cbrobotica.org/?page_id=6&lang=en
2. COPOT, Cosmin, IONESCU, Clara and KEYSER, Robin De. *Interdisciplinary project-based learning at master level: control of robotic mechatronic systems*. In : *IFAC-PapersOnLine*. Elsevier B.V., 2016. p. 314–319.
3. KOZÁK, Štefan. *Multidisciplinary Approach and Dual Education in Control Engineering for Mechatronics*. In : *IFAC-PapersOnLine*. Elsevier B.V., 2016. p. 52–56.
4. *15th International Workshop on Research and Education in Mechatronics*. IEEE, 2014. ISBN 9781479930296.
5. *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics : Suntec Convention and Exhibition Center, Singapore, July 14-17, 2009*. IEEE, 2009. ISBN 9781424428533.
6. ZIMENKO, Konstantin, BAZYLEV, Dmitry, MARGUN, Alexey and KREMLEV, Artem. *Application of Innovative Mechatronic Systems in Automation and Robotics Learning*. [no date].
7. SPERANZA NETO, Mauro, ALBUQUERQUE, Allan Nogueira de and ASSAD, Marília Maurell. *Small Scale Mechatronics Devices as Educational and Research Engineering Tools*. In : *IFAC-PapersOnLine*. Elsevier B.V., 2016. p. 248–255.
8. TLALE, N S and ZHANG, P. *Teaching the Design of Parallel Manipulators and Their Controllers Implementing MATLAB, Simulink, SimMechanics and CAD**. [no date].
9. RAOUFI, Kamyar, HILBERT, Alejandra, SANCHEZ, Christopher A, FAN, Zhaoyan, SENCER, Burak, STEINGRIMSSON, Baldur, JOHNSTON, Matthew L and HAAPALA, Karl R. *Manufacturing Letters Development of a Quantitative Learning Assessment (QuLA) Method and its Application in Manufacturing Engineering Courses in Mechatronics [*online*]*. 2025. Available from: www.sciencedirect.com
10. EBERT-UPHOFF, Imme. *Introducing Parallel Manipulators Through Laboratory Experiments*. [no date].
11. VENKAT RAAMAN, M., ARAVIND, S., PAVEL, R., KUPPAN CHETTY, R. M. and DHANRAJ, Joshuva Arockia. *Design and Development of a General-Purpose Low-Cost Stewart Platform for Laboratory Teaching: A Mechatronics Approach*. In : *Lecture Notes in Mechanical Engineering*. Springer Science and Business Media Deutschland GmbH, 2021. p. 469–479. ISBN 9789811609411.
12. KLIMES, D., RIPEL, T., SURANSKY, M. and VEJLUPEK, J. *The design and use of 3DOF manipulator as a platform for education in mechatronics*. In : *Mechatronics 2013: Recent Technological and Scientific Advances*. Springer Science and Business Media Deutschland GmbH, 2014. p. 877–882. ISBN 9783319022932.
13. MANZOOR, Sarah, UL ISLAM, Raza, KHALID, Aayman, SAMAD, Abdul and IQBAL, Jamshed. *An open-source multi-DOF articulated robotic educational platform for autonomous object manipulation. Robotics and Computer-Integrated Manufacturing*. June 2014. Vol. 30, no. 3, p. 351–362. DOI 10.1016/j.rcim.2013.11.003.
14. BALBUENA, Jose, SINCHE, Julio, QUIROZ, Diego, ARCE, Diego and VILLOTA, Elizabeth. *PlatROB: An open-source, modular, and low-cost hardware platform for mobile robotics and AI education. [*online*]*. 2026. DOI 10.17605/OSF.IO/KU92V. Available from: https://doi.org/10.17605/OSF.IO/KU92V

15. *2017 International Conference on Information and Digital Technologies*. IEEE, 2017. ISBN 9781509056897.
16. *TENCON 2010, 2010 IEEE Region 10 Conference : date, 21-24 Nov. 2010, Fukuoka, Japan*. [IEEE], 2010. ISBN 9781424468904.
17. BEUCHAT, Paul N., BRADFORD, Glenn J. and BUSKES, Gavin. *Challenges and opportunities of using differential-drive robots with project-based learning pedagogies*. In : *IFAC-PapersOnLine*. Elsevier B.V., 1 July 2022. p. 186–193.
18. FU, Lujun, GONG, Youmin, DONG, Haiting and MA, Guangfu. *Exploring the Teaching Mechanism for Developing Engineering Innovation Abilities through Project-Based Learning in Spacecraft Control*. In : *IFAC-PapersOnLine*. Elsevier B.V., 1 August 2025. p. 1772–1775.
19. EICHLER, Annika, HOFFMANN, Christian, KAUTZ, Christian and WERNER, Herbert. *Design of tutorial activities and homework assignments for a large-enrollment introductory course in control systems*. In : *IFAC Proceedings Volumes (IFAC-PapersOnline)*. IFAC Secretariat, 2013. p. 43–48. ISBN 9783902823434.
20. IQBAL, Jamshed, RIAZ UN NABI, S, ATTAYYAB KHAN, Abdul and KHAN, Hamza. *A Novel Track-Drive Mobile Robotic Framework for Conducting Projects on Robotics and Control Systems [*online*]*. 2013. Available from: http://www.lifesciencesite.comhttp://www.lifesciencesite.comhttp://www.lifesciencesite.com21
21. CINUN, Benedictus C.G., TAMBA, Tua A. and WAHAB, Faisal. *Kinematic Modeling and Prototype Development of a Six Degrees-of-Freedom Stewart Platform*. In : *2024 14th International Conference on System Engineering and Technology, ICSET 2024 - Proceeding*. Institute of Electrical and Electronics Engineers Inc., 2024. p. 143–147. ISBN 9798331504526.
22. EBERT-UPHOFF, Imme. *Introducing Parallel Manipulators Through Laboratory Experiments*. [no date].
23. *2003 4th International Conference on Control and Automation Proceedings*. IEEE, 2003. ISBN 078037777X.
24. HAJIMIRZAALIAN, Hamidreza, MOOSAVI, Hasan and MASSAH, Mehdi. *Analyzing and simulating the inverse and the direct dynamics of parallel robot Stewart platform*. In : *2nd International Conference on Computer and Network Technology, ICCNT 2010*. 2010. p. 136–141. ISBN 9780769540429.
25. DATTA, Samiran, DAS, Akash and GAYEN, Rintu Kumar. *Kinematic Analysis of Stewart Platform using MATLAB*. In : *2021 5th International Conference on Electronics, Materials Engineering and Nano-Technology, IEMENTech 2021*. Institute of Electrical and Electronics Engineers Inc., 2021. ISBN 9781665418034.
26. KARAKUŞ, Raşit and TANIK, Çağıl Merve. *Compliant universal joint with preformed flexible segments. Journal of Mechanical Science and Technology [*online*]*. Vol. 36, no. 3, p. 2022. DOI 10.1007/s12206-022-04. Available from: http://doi.org/10.1007/s12206-022-1026-5
27. *2011 6th IEEE Conference on Industrial Electronics and Applications*. IEEE, 2011. ISBN 9781424487561.
28. LA, Qiyun, ZHOU, Xubin and LYE, Wang. *Vibration control study of large deployable reflector antenna based on Stewart platform**. [no date].
29. *Proceedings, 2019 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET) : October 23-24, 2019, Tangerang, Indonesia*. IEEE, 2019. ISBN 9781728124827.
30. *2015 International Conference Stability and Control Processes in Memory of V.I. Zubov*. IEEE, 2015. ISBN 9781467376983.

31. TMG, Yung, CHEN, Yu-Shin, JAR, Ho-Chin and KANG, Yuan. *Pmceedingr Of the 2004 IEEE lntematlonal Conference on Robatiu L Automation Modeling and Control for A Gough-Stewart Platform CNC Machine*. 2004.
32. Alan Buterbaughl, Brian M. Kent, Carl Mentzer, Mark Scott, and William Forster. *Demonstration of an Inverted Stewart Platform Target-S us pension System Using Lightweight, High-Tensile Strings*, In: IEEE Antennas and Propagation Magazine, Vol. 49, 2007
33. ALI, Zeeshan, WAQAS, Mudasser, WAQAS, Muzzamil, AFTAB, Usama and RAZA, Ali. *Design and Development of Stewart Platform To Balance the Ball Using Fuzzy Logic Control*. In : *2022 International Conference on IT and Industrial Technologies, ICIT 2022*. Institute of Electrical and Electronics Engineers Inc., 2022. ISBN 9781665489454.

Appendices

Appendix 1. Arduino Code

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_PWMServoDriver.h>

// ================ PCA9685 ================
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
#define SERVOMIN 150
#define SERVOMAX 600
uint8_t servos[6] = {15,14,13,12,11,10};
float currentAngles[6] = {90,90,90,90,90,90};
float targetAngles[6]  = {90,90,90,90,90,90};

// ================ MPU6500 (SPI) ================
#define CS_PIN 10
float AccX, AccY, AccZ;
float GyroZ;
float roll, pitch, yaw = 0;
unsigned long prevTime = 0;
unsigned long lastIMUPrint = 0;

// ================ SERIAL ================
char inputBuffer[50];
byte bufferIndex = 0;

// ================ SETUP ================
void setup() {

  Serial.begin(115200);

  // I2C for PCA9685
  Wire.begin();
  Wire.setWireTimeout(3000, true);
  pwm.begin();
  pwm.setPWMFreq(50);

  // SPI for MPU6500
  SPI.begin();
  SPI.beginTransaction(SPISettings(1000000, MSBFIRST, SPI_MODE0)); // 1 MHz stable
  pinMode(CS_PIN, OUTPUT);
  digitalWrite(CS_PIN, HIGH);
  initMPU();
  prevTime = millis();
}

// ================ LOOP ================
void loop() {
```

```
  readSerialNonBlocking();
  moveAllServosSmooth();
  // readIMU();

  // if (millis() - lastIMUPrint > 50) {
  //   printIMU();
  //   lastIMUPrint = millis();
  // }

  // delayMicroseconds(200); // prevent bus overload
}

// ================= SERIAL =================
void readSerialNonBlocking() {

  while (Serial.available()) {

    char c = Serial.read();

    if (c == '\n') {

      inputBuffer[bufferIndex] = '\0';
      parseInput(inputBuffer);
      bufferIndex = 0;

    } else {

      if (bufferIndex < 49) {
        inputBuffer[bufferIndex++] = c;
      }
    }
  }
}

void parseInput(char* input) {

  int values[6];
  int i = 0;

  char *token = strtok(input, " ");

  while (token != NULL && i < 6) {
    values[i++] = atoi(token);
    token = strtok(NULL, " ");
  }

  if (i == 6) {

    for (int j = 0; j < 6; j++) {

      int inputVal = constrain(values[j], -35, 35);
```

```
      targetAngles[j] = map(inputVal, -35, 35, 55, 125);
    }
  }
}

// ================= SMOOTH SERVO =================
void moveAllServosSmooth() {

  static unsigned long lastStepTime = 0;
  const int stepInterval = 10; // ms

  if (millis() - lastStepTime < stepInterval) return;

  lastStepTime = millis();

  for (int i = 0; i < 6; i++) {

    float error = targetAngles[i] - currentAngles[i];

    if (abs(error) > 0.1) {
      currentAngles[i] += error * 0.2;  // smoothing factor
    }

    int angle = (int)currentAngles[i];

    if (i % 2 == 0) angle = 180 - angle;

    setServoAngle(servos[i], angle);
  }
}

void setServoAngle(uint8_t servo, int angle) {

  int pulse = map(angle, 0, 180, SERVOMIN, SERVOMAX);
  pwm.setPWM(servo, 0, pulse);
}

// ================= MPU INIT =================
void initMPU() {

  digitalWrite(CS_PIN, LOW);
  SPI.transfer(0x6B);
  SPI.transfer(0x00);
  digitalWrite(CS_PIN, HIGH);

  delay(100);
}

// ================= IMU READ (FIXED) =================
void readIMU() {
```

```
  digitalWrite(CS_PIN, LOW);

  SPI.transfer(0x3B | 0x80);  // start read

  int16_t ax = (SPI.transfer(0x00) << 8) | SPI.transfer(0x00);
  int16_t ay = (SPI.transfer(0x00) << 8) | SPI.transfer(0x00);
  int16_t az = (SPI.transfer(0x00) << 8) | SPI.transfer(0x00);

  SPI.transfer(0x00); SPI.transfer(0x00); // temp

  SPI.transfer(0x00); SPI.transfer(0x00); // gyro X
  SPI.transfer(0x00); SPI.transfer(0x00); // gyro Y

  int16_t gz = (SPI.transfer(0x00) << 8) | SPI.transfer(0x00);

  digitalWrite(CS_PIN, HIGH);

  AccX = ax / 16384.0;
  AccY = ay / 16384.0;
  AccZ = az / 16384.0;

  GyroZ = gz / 131.0;

  unsigned long currentTime = millis();
  float dt = (currentTime - prevTime) / 1000.0;
  prevTime = currentTime;

  roll  = atan2(AccY, AccZ) * 180 / PI;
  pitch = atan2(-AccX, sqrt(AccY * AccY + AccZ * AccZ)) * 180 / PI;

  yaw += GyroZ * dt;
}

// ================ PRINT ================
void printIMU() {

  Serial.print("R:");
  Serial.print(roll,1);
  Serial.print(" P:");
  Serial.print(pitch,1);
  Serial.print(" Y:");
  Serial.println(yaw,1);
}
```

Appendix 2. Python Code (GUI)

```
import numpy as np
import math
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import tkinter as tk
import serial
import time
from serial.tools import list_ports

def find_arduino_port():
    ports = list_ports.comports()
    for port in ports:
        desc = port.description.lower()
        if "arduino" in desc or "ch340" in desc or "usb serial" in desc:
            print(f"Using port: {port.device}")
            return port.device
    raise Exception("Arduino not found")

# ================ IK ================
class InverseKinematics:

    def __init__(self):
        self.Base_points = np.array([
            [-87.11, -93.02, 0], [87.11, -93.02, 0], [124.11, -28.93, 0],
            [37, 121.95, 0], [-37, 121.95, 0], [-124.11, -28.93, 0]
        ])

        self.Platform_points = np.array([
            [-39, -128.19, 0], [39, -128.19, 0], [130.52, 30.32, 0],
            [91.52, 97.87, 0], [-91.52, 97.87, 0], [-130.52, 30.32, 0]
        ])

    def rotation_matrix(self, Roll, Pitch, Yaw):

        Rx = np.array([[1,0,0],[0,np.cos(Roll),-
np.sin(Roll)],[0,np.sin(Roll),np.cos(Roll)]])
        Ry = np.array([[np.cos(Pitch),0,np.sin(Pitch)],[0,1,0],[-
np.sin(Pitch),0,np.cos(Pitch)]])
        Rz = np.array([[np.cos(Yaw),-
np.sin(Yaw),0],[np.sin(Yaw),np.cos(Yaw),0],[0,0,1]])

        return Rx @ Ry @ Rz

    def platform_points_calculator(self,x,y,z,Roll,Pitch,Yaw):

        Roll,Pitch,Yaw = np.radians([Roll,Pitch,Yaw])
        R = self.rotation_matrix(Roll,Pitch,Yaw)
```

```
        rotated = (R @ self.Platform_points.T).T
        transformed = rotated + np.array([x,y,z])

        distances = np.linalg.norm(self.Base_points - transformed,axis=1)

        return distances, transformed

    def calculate_servo_angles(self,link_lengths,return_degrees=False):

        a = link_lengths
        b = 35.51
        c = 161

        theta_norm = 90.42

        angles = []
        thetas = []

        for i in range(6):

            cos_theta = (a[i]**2 + b**2 - c**2) / (2*a[i]*b)
            cos_theta = np.clip(cos_theta,-1.0,1.0)

            theta = np.arccos(cos_theta)

            if return_degrees:
                thetas.append(np.degrees(theta))
                theta = theta_norm - np.degrees(theta)

            angles.append(theta)

        return np.array(angles), thetas

    def plot_stewart_platform(self,ax,base_points,platform_points,triangle_angles):

        ax.clear()

        b = 35.51

        base_points = np.array(base_points)
        platform_points = np.array(platform_points)

        ax.scatter(base_points[:,0],base_points[:,1],base_points[:,2],color='cyan',
s=60)
        ax.plot(*np.vstack([base_points,base_points[0]]).T,color='deepskyblue')

        ax.scatter(platform_points[:,0],platform_points[:,1],platform_points[:,2],c
olor='orange',s=60)
        ax.plot(*np.vstack([platform_points,platform_points[0]]).T,color='orangered
')
```

```
# ===== Correct servo pairing =====
servo_pairs = [
    (0, 5),  # 1 & 6
    (1, 2),  # 2 & 3
    (3, 4)   # 4 & 5
]

pair_dirs = []
for p1, p2 in servo_pairs:
    vec = base_points[p2] - base_points[p1]
    pair_dirs.append(vec / np.linalg.norm(vec))

pair_map = {}
for idx, (p1, p2) in enumerate(servo_pairs):
    pair_map[p1] = idx
    pair_map[p2] = idx

for i in range(6):

    bp = base_points[i]
    pp = platform_points[i]

    leg = pp - bp
    leg_len = np.linalg.norm(leg)
    leg_dir = leg / leg_len

    pair_index = pair_map[i]
    servo_axis = pair_dirs[pair_index]

    # perpendicular direction
    u = servo_axis - np.dot(servo_axis,leg_dir)*leg_dir

    # keep servo arm horizontal
    u[2] = 0

    u = u / np.linalg.norm(u)

    # flip within each pair
    if i in [5, 2, 4]:  # second of each pair
        u = -u

    # global flip (change sign if needed)
    u = -u

    theta = np.radians(triangle_angles[i])

    elbow = bp + b*np.cos(theta)*leg_dir + b*np.sin(theta)*u

    # 🟢 Servo arm
    ax.plot([bp[0],elbow[0]],
            [bp[1],elbow[1]],
```

```
                    [bp[2],elbow[2]],
                    color='lime',linewidth=3)

            # 🟢 Linkage
            ax.plot([elbow[0],pp[0]],
                    [elbow[1],pp[1]],
                    [elbow[2],pp[2]],
                    color='magenta',linewidth=3)

            # dashed leg
            ax.plot([bp[0],pp[0]],
                    [bp[1],pp[1]],
                    [bp[2],pp[2]],
                    linestyle='--',color='white')

        ax.set_xlim([-200,200])
        ax.set_ylim([-200,200])
        ax.set_zlim([0,250])
        ax.set_facecolor('#111111')

# ================ GUI ================
class StewartGUI:

    def __init__(self):

        self.Ik = InverseKinematics()
        self.Platform_height = 145

        port = find_arduino_port()
        self.ser = serial.Serial(port, 115200, timeout=1)
        time.sleep(2)

        self.root = tk.Tk()
        self.root.title("Stewart Platform Simulator")
        self.root.configure(bg="#121212")

        self.root.columnconfigure(0, weight=4)
        self.root.columnconfigure(1, weight=1)
        self.root.rowconfigure(1, weight=1)

        plt.style.use("dark_background")

        self.fig = plt.figure(figsize=(8,6))
        self.ax = self.fig.add_subplot(111, projection='3d')

        self.canvas = FigureCanvasTkAgg(self.fig, master=self.root)
        self.canvas.get_tk_widget().grid(row=0, column=0, sticky="nsew")

        # ===== Servo angle display =====
        self.servo_label = tk.Label(
            self.root,
```

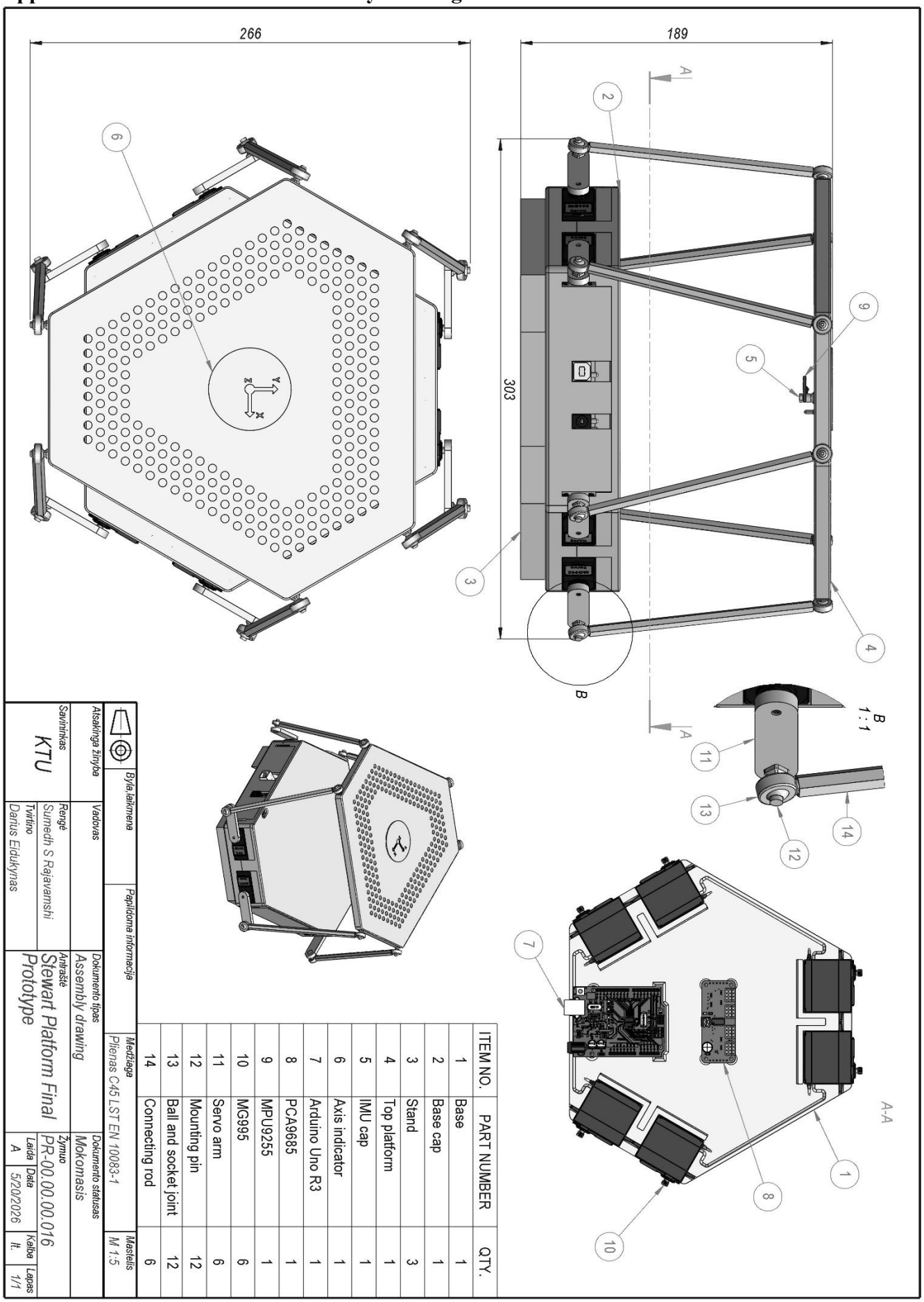
```
            text="Servo angles:",
            bg="#121212",
            fg="white",
            font=("Consolas", 11)
        )
        self.servo_label.grid(row=1, column=0, sticky="ew", padx=10, pady=5)

        control_frame = tk.Frame(self.root, bg="#1e1e1e", padx=10, pady=10)
        control_frame.grid(row=0, column=1, sticky="nsew")

        self.sliders = {}
        self.oscillating = False
        self.t = 0

        # ===== Axis limits =====
        self.axis_limits = {
            "X":40, "Y":40, "Z":20,
            "Roll":10, "Pitch":10, "Yaw":20
        }

        slider_specs = [
            ("X",-40,40), ("Y",-40,40), ("Z",-20,20),
            ("Roll",-10,10), ("Pitch",-10,10), ("Yaw",-20,20)
        ]

        for name,minv,maxv in slider_specs:
            tk.Label(control_frame, text=name, bg="#1e1e1e",
fg="white").pack(anchor="w")
            slider = tk.Scale(control_frame, from_=minv, to=maxv,
                              orient=tk.HORIZONTAL, resolution=0.5,
                              command=self.update_platform,
                              bg="#1e1e1e", fg="white",
                              troughcolor="#444")
            slider.pack(fill="x", pady=3)
            self.sliders[name] = slider

        # ===== Axis 1 =====
        tk.Label(control_frame, text="Axis 1", bg="#1e1e1e",
fg="white").pack(anchor="w")

        self.axis1_var = tk.StringVar(value="X")
        tk.OptionMenu(control_frame, self.axis1_var,
                      "X","Y","Z","Roll","Pitch","Yaw",
                      command=self.update_amp1_range).pack(fill="x")

        tk.Label(control_frame, text="Amplitude 1", bg="#1e1e1e",
fg="white").pack(anchor="w")

        self.amp1_slider = tk.Scale(control_frame, from_=1, to=60,
                                    orient=tk.HORIZONTAL,
                                    bg="#1e1e1e", fg="white",
```

```
                                troughcolor="#444")
        self.amp1_slider.pack(fill="x")

        # ===== Axis 2 =====
        tk.Label(control_frame, text="Axis 2", bg="#1e1e1e",
fg="white").pack(anchor="w")

        self.axis2_var = tk.StringVar(value="None")
        tk.OptionMenu(control_frame, self.axis2_var,
                      "None","X","Y","Z","Roll","Pitch","Yaw",
                      command=self.update_amp2_range).pack(fill="x")

        tk.Label(control_frame, text="Amplitude 2", bg="#1e1e1e",
fg="white").pack(anchor="w")

        self.amp2_slider = tk.Scale(control_frame, from_=1, to=60,
                                    orient=tk.HORIZONTAL,
                                    bg="#1e1e1e", fg="white",
                                    troughcolor="#444")
        self.amp2_slider.pack(fill="x")

        # ===== Phase =====
        tk.Label(control_frame, text="Phase Shift (deg)", bg="#1e1e1e",
fg="white").pack(anchor="w")

        self.phase_slider = tk.Scale(control_frame, from_=0, to=90,
                                     orient=tk.HORIZONTAL,
                                     bg="#1e1e1e", fg="white",
                                     troughcolor="#444")
        self.phase_slider.pack(fill="x")

        # ===== Speed =====
        tk.Label(control_frame, text="Speed (Hz)", bg="#1e1e1e",
fg="white").pack(anchor="w")

        self.speed_slider = tk.Scale(control_frame, from_=0.1, to=10.0,
                                     resolution=0.1,
                                     orient=tk.HORIZONTAL,
                                     bg="#1e1e1e", fg="white",
                                     troughcolor="#444")
        self.speed_slider.set(1.0)
        self.speed_slider.pack(fill="x")

        # ===== Buttons =====
        tk.Button(control_frame, text="Start Oscillation",
                  command=self.start_oscillation,
                  bg="#2e7d32", fg="white").pack(fill="x", pady=5)

        tk.Button(control_frame, text="Stop Oscillation",
                  command=self.stop_oscillation,
                  bg="#c62828", fg="white").pack(fill="x", pady=5)
```

```
        tk.Button(control_frame, text="Reset",
                  command=self.reset_values,
                  bg="#444", fg="white").pack(fill="x", pady=5)

        tk.Button(control_frame, text="Exit",
                  command=self.root.destroy,
                  bg="#880e1c", fg="white").pack(fill="x", pady=5)

        # ☑ Initialize correct ranges
        self.update_amp1_range()
        self.update_amp2_range()

        self.update_platform(None)
        self.root.mainloop()

    # ===== Dynamic amplitude limits =====
    def update_amp1_range(self, event=None):
        axis = self.axis1_var.get()
        max_val = self.axis_limits[axis]
        self.amp1_slider.config(to=max_val)

    def update_amp2_range(self, event=None):
        axis = self.axis2_var.get()
        if axis == "None":
            self.amp2_slider.config(to=1)
        else:
            max_val = self.axis_limits[axis]
            self.amp2_slider.config(to=max_val)

    def reset_values(self):
        for slider in self.sliders.values():
            slider.set(0)
        self.update_platform(None)

    # ===== Oscillation =====
    def start_oscillation(self):
        self.oscillating = True
        self.t = 0
        self.run_oscillation()


    def stop_oscillation(self):
        self.oscillating = False

    def run_oscillation(self):
        if not self.oscillating:
            return

        freq = self.speed_slider.get()
```

```
            # Axis 1
            axis1 = self.axis1_var.get()
            amp1 = min(self.amp1_slider.get(), self.axis_limits[axis1])
            val1 = amp1 * math.sin(self.t)
            self.sliders[axis1].set(val1)

            # Axis 2
            axis2 = self.axis2_var.get()
            if axis2 != "None":
                amp2 = min(self.amp2_slider.get(), self.axis_limits[axis2])
                phase = math.radians(self.phase_slider.get())
                val2 = amp2 * math.sin(self.t + phase)
                self.sliders[axis2].set(val2)

            self.t += 2 * math.pi * freq * 0.02

            self.root.after(20, self.run_oscillation)

        # ===== Main update =====
        def update_platform(self, event):

            X = self.sliders["X"].get()
            Y = self.sliders["Y"].get()
            Z = self.sliders["Z"].get()

            Pitch = self.sliders["Roll"].get()
            Roll = self.sliders["Pitch"].get()
            Yaw = self.sliders["Yaw"].get()

            link_lengths, platform_points = self.Ik.platform_points_calculator(
                X, Y, Z + self.Platform_height, Roll, Pitch, Yaw
            )

            angles, theta = self.Ik.calculate_servo_angles(link_lengths,
    return_degrees=True)

            # ===== Update servo angle display =====
            text = "   ".join([f"S{i+1}:{angles[i]:.1f}°" for i in range(6)])
            self.servo_label.config(text=text)
            serial_data = " ".join(str(int(x)) for x in np.round(angles)) + "\n"
            self.ser.write(serial_data.encode())

            self.Ik.plot_stewart_platform(
                self.ax,
                self.Ik.Base_points,
                platform_points,
                theta
            )

            self.canvas.draw()
    StewartGUI()
```

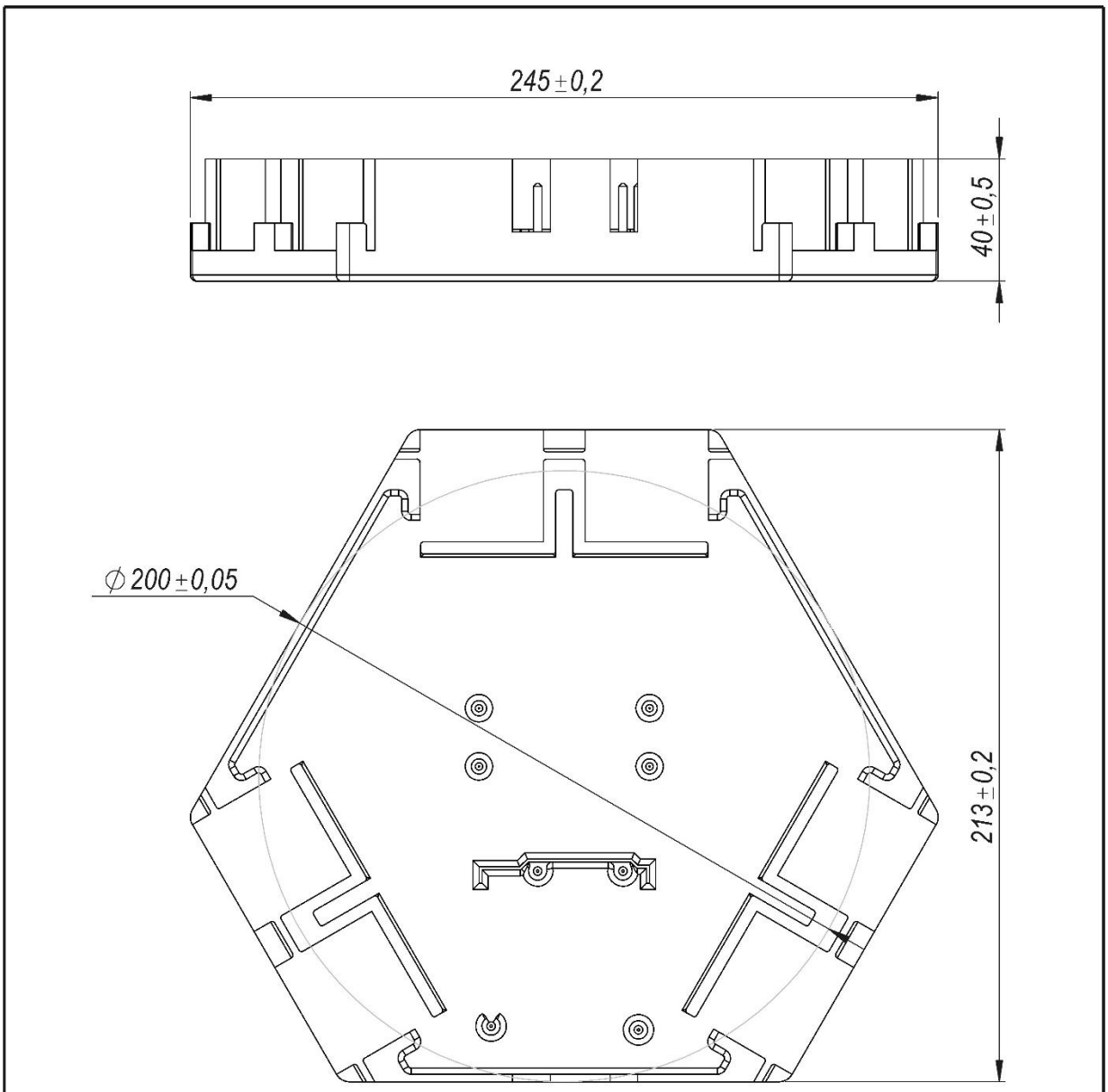
Appendix 3. Stewart Platform assembly drawing

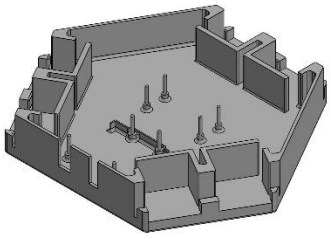


ITEM NO.	PART NUMBER	QTY.
1	Base	1
2	Base cap	1
3	Stand	3
4	Top platform	1
5	IMU cap	1
6	Axis indicator	1
7	Arduino Uno R3	1
8	PCA9685	1
9	MPU9255	1
10	MG995	6
11	Servo arm	6
12	Mounting pin	12
13	Ball and socket joint	12
14	Connecting rod	6

Atsakinga žinyba	Byla,laikmena	Medžiaga	Dokumento statusas	Mastelis
Vadovas	Papildoma informacija	Plienas C45 LST EN 10083-1	Mokomasis	M 1:5
Rengė Sumedh S Rajavamshi	Dokumento tipas Assembly drawing		Žymuo PR-00.00.00.016	
Tvirtino Darius Eidukynas	Antraštė Stewart Platform Final Prototype		Laida A	Data 5/20/2026
Savininkas KTU			Katilka	Lapas 1/1

Appendix 4. Base

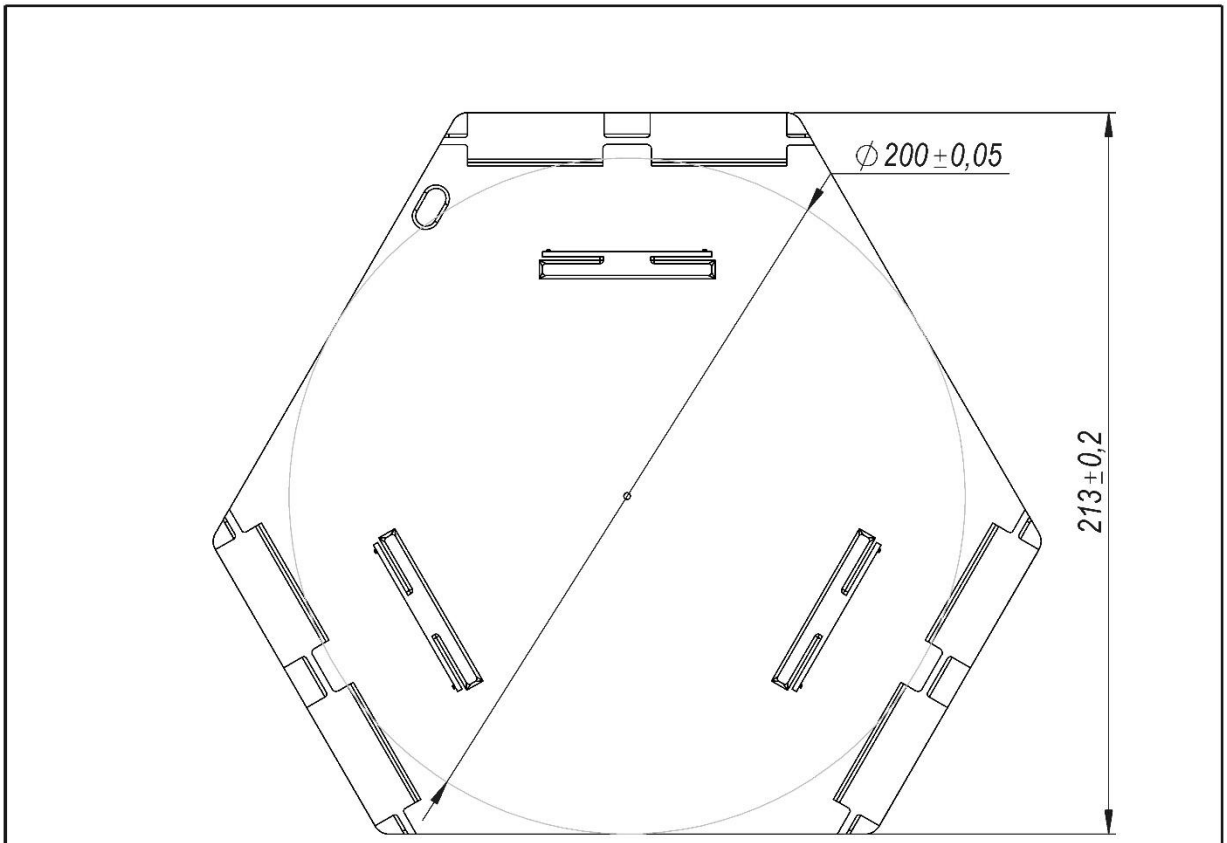


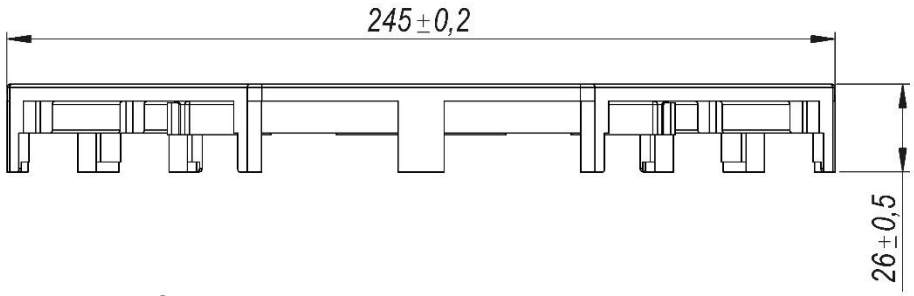


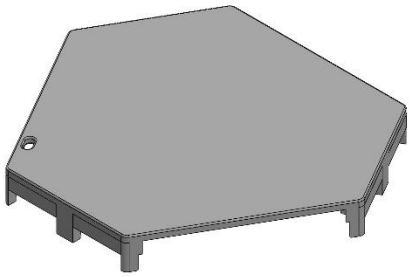
	Byla,laikmena	Papildoma informacija	Medžiaga Plienas C45 LST EN 10083-1	Mastelis M 1:5

Atsakinga žinyba	Vadovas	Dokumento tipas Part drawing	Dokumento statusas Mokomasis
Savininkas KTU	Rengė Sumedh S Rajavamshi	Antraštė Base	Žymuo PR-00.00.00.016
	Tvirtino Darius Eidukynas		Laida: A; Data: 5/20/2026; Kalba: lt.; Lapas: 1/1

Appendix 5. Base cap

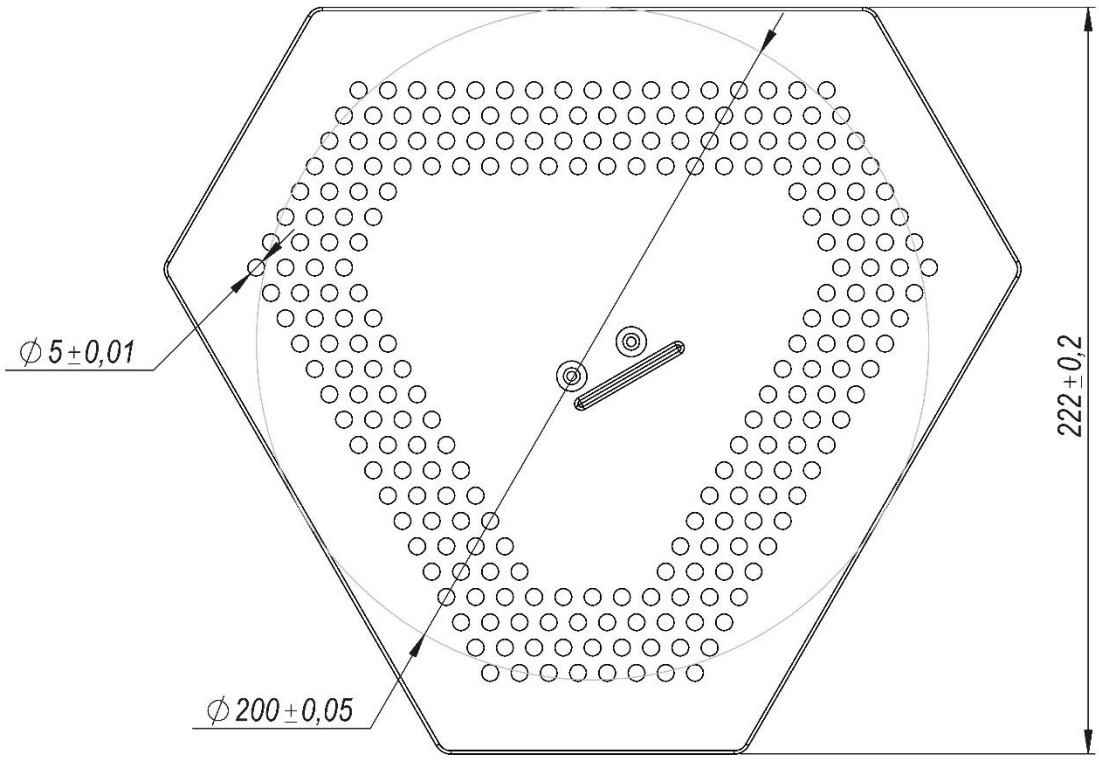


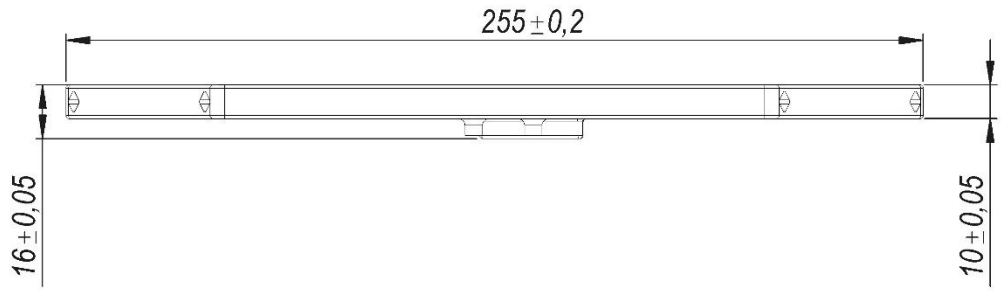


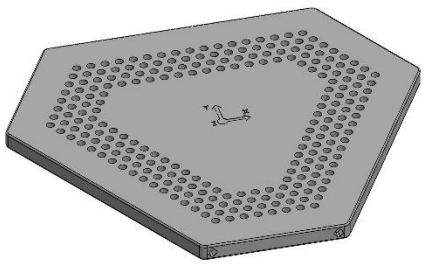


	Byla,laikmena	Papildoma informacija	Medžiaga: Plienas C45 LST EN 10083-1	Mastelis: M 1:5
Atsakinga žinyba	Vadovas	Dokumento tipas: Part drawing	Dokumento statusas: Mokomasis	
Savininkas: KTU	Rengė: Sumedh S Rajavamshi	Antraštė: Base cap	Žymuo: PR-00.00.00.016	
	Tvirtino: Darius Eidukynas		Laida: A / Data: 5/20/2026 / Kalba: lt. / Lapas: 1/1	

Appendix 6. Top platform





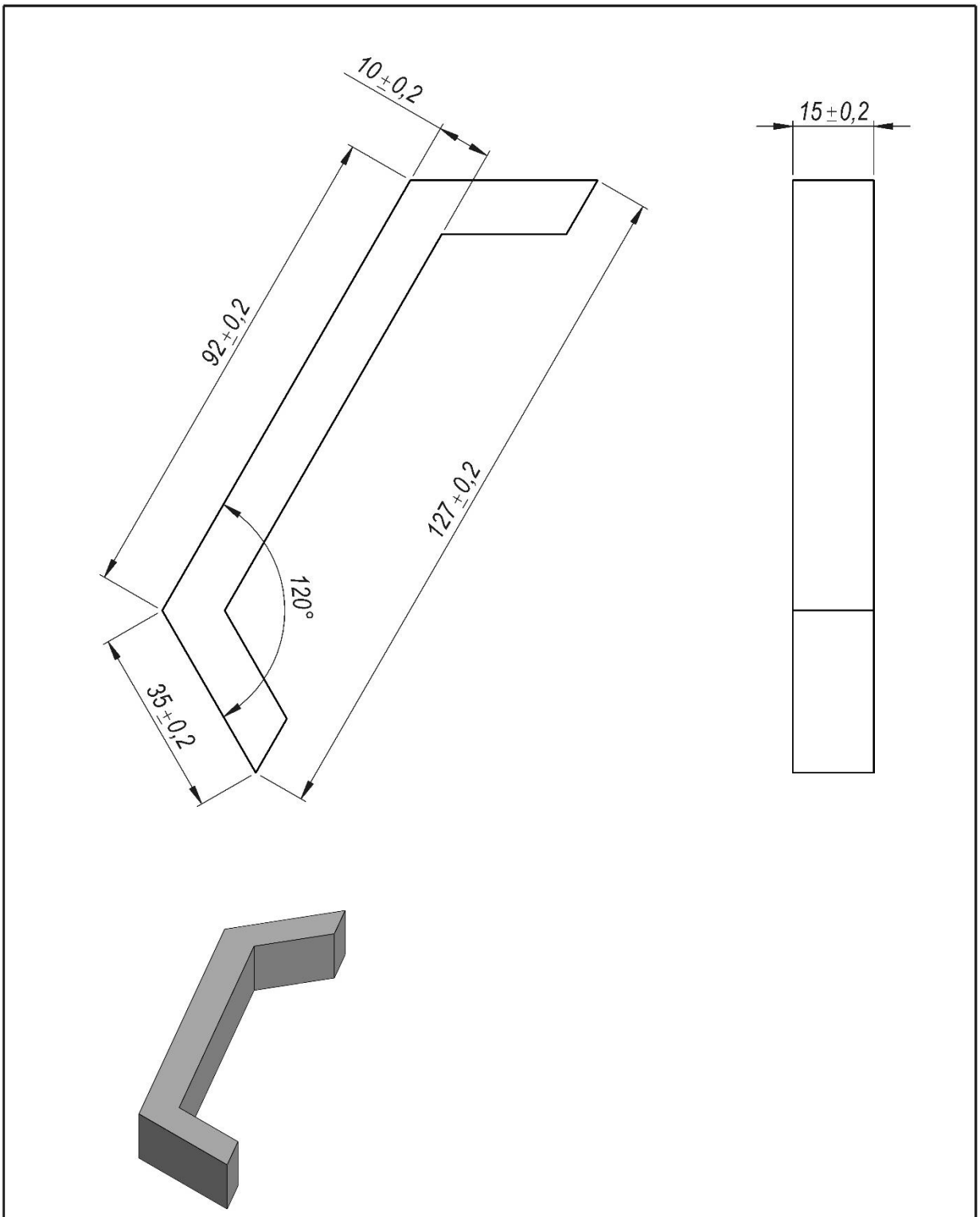


	Byla,laikmena	Papildoma informacija	Medžiaga	Mastelis
			Plienas C45 LST EN 10083-1	M 1:5

Atsakinga žinyba	Vadovas	Dokumento tipas	Dokumento statusas
		Part drawing	Mokomasis

Savininkas	Rengė	Antraštė	Žymuo			
KTU	Sumedh S Rajavamshi	Top platform	PR-00.00.00.016			
	Tvirtino		Laida	Data	Kalba	Lapas
	Darius Eidukynas		A	5/20/2026	lt.	1/1

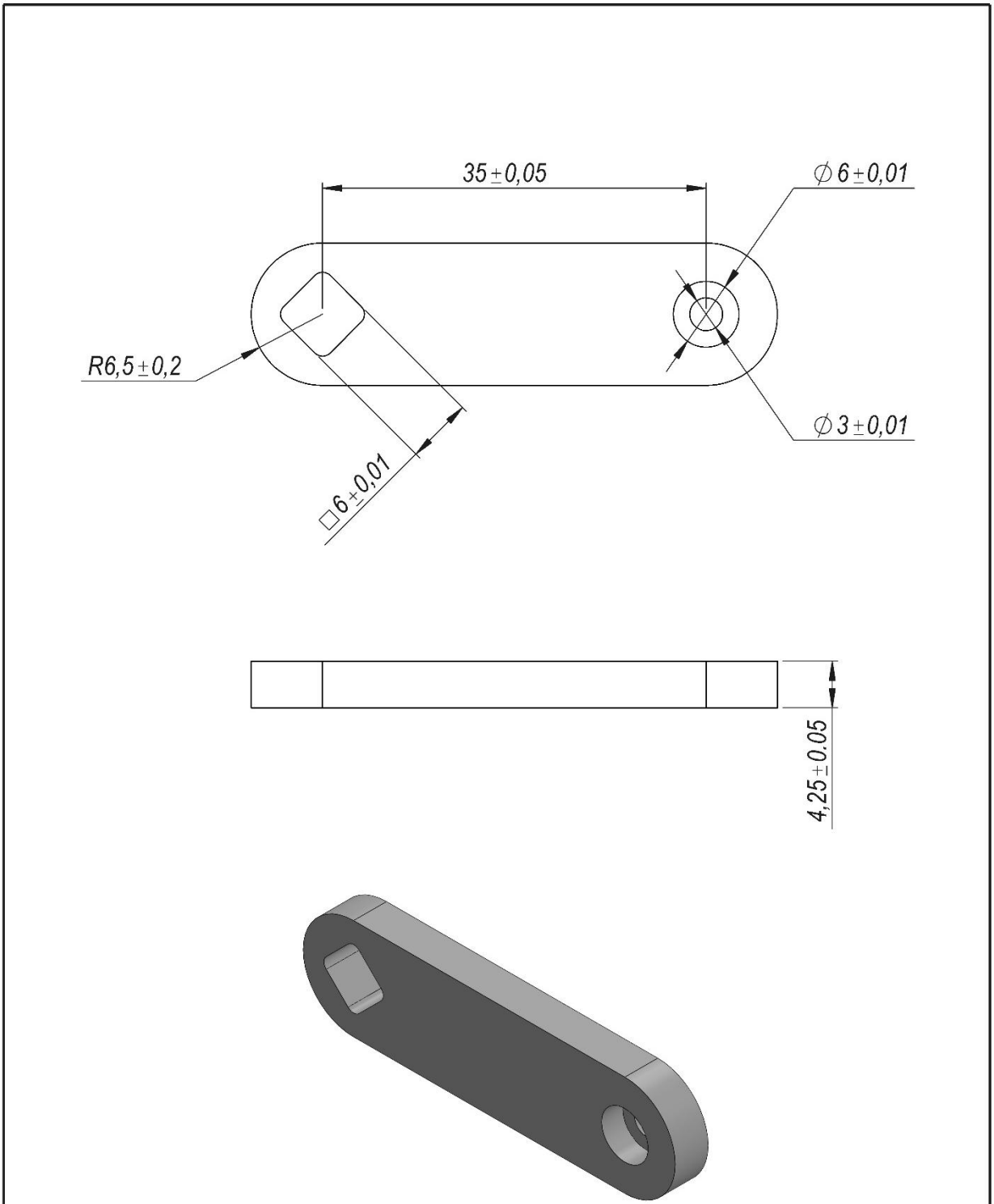
Appendix 7. Stand



	Byla,laikmena	Papildoma informacija	Medžiaga: Plienas C45 LST EN 10083-1	Mastelis: M 1:2

Atsakinga žinyba	Vadovas	Dokumento tipas: Part drawing	Dokumento statusas: Mokomasis		
Savininkas: KTU	Rengė: Sumedh S Rajavamshi	Antraštė: Stand	Žymuo: PR-00.00.00.016		
	Tvirtino: Darius Eidukynas		Laida: A	Data: 5/20/2026	Kalba: lt. / Lapas: 1/1

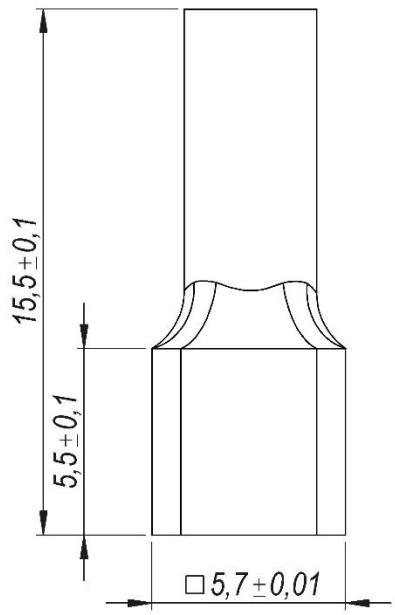
Appendix 8. Servo arm

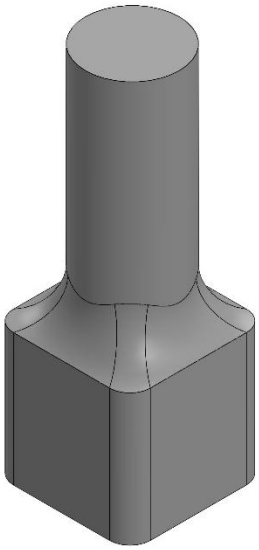


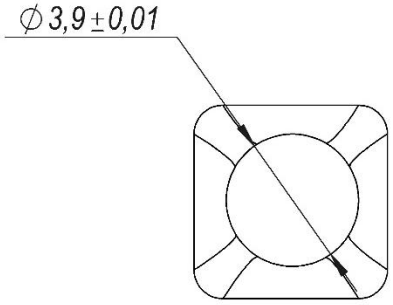
	Byla,laikmena	Papildoma informacija	Medžiaga Plienas C45 LST EN 10083-1	Mastelis M 2:1

Atsakinga žinyba	Vadovas	Dokumento tipas Part drawing	Dokumento statusas Mokomasis			
Savininkas KTU	Rengė Sumedh S Rajavamshi	Antraštė Servo arm	Žymuo PR-00.00.00.016			
	Tvirtino Darius Eidukynas		Laida A	Data 5/20/2026	Kalba lt.	Lapas 1/1

Appendix 9. Mounting pin



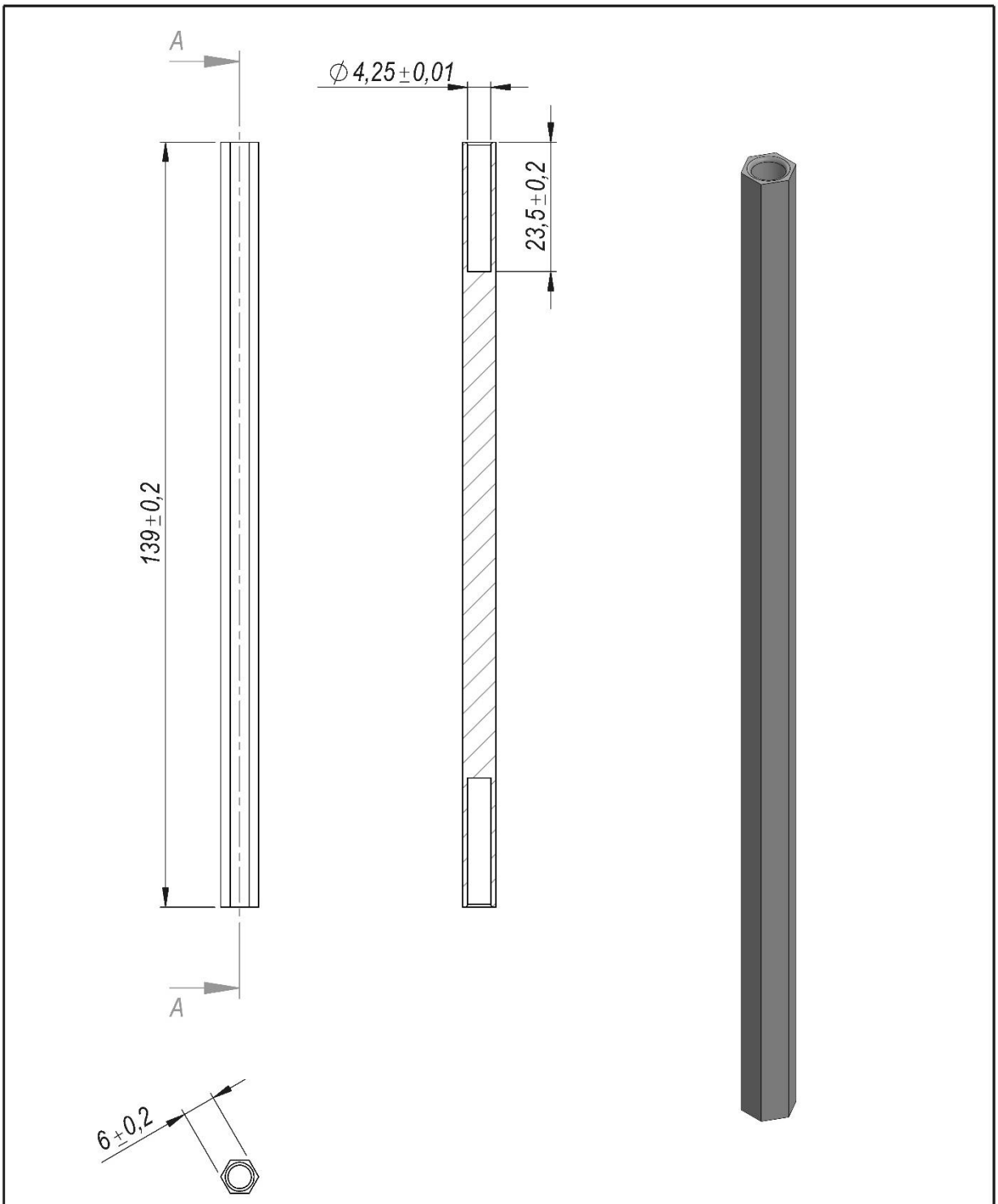




	Byla,laikmena	Papildoma informacija	Medžiaga: Plienas C45 LST EN 10083-1	Mastelis: M 5:1

Atsakinga žinyba	Vadovas	Dokumento tipas: Part drawing	Dokumento statusas: Mokomasis
Savininkas: KTU	Rengė: Sumedh S Rajavamshi	Antraštė: Mounting pin	Žymuo: PR-00.00.00.016
	Tvirtino: Darius Eidukynas		Laida: A; Data: 5/20/2026; Kalba: lt.; Lapas: 1/1

Appendix 10. Connecting rod



	Byla,laikmena	Papildoma informacija	Medžiaga: Plienas C45 LST EN 10083-1	Mastelis: M 1:1
Atsakinga žinyba	Vadovas	Dokumento tipas: Part drawing	Dokumento statusas: Mokomasis	
Savininkas: KTU	Rengė: Sumedh S Rajavamshi Tvirtino: Darius Eidukynas	Antraštė: Connecting rod	Žymuo: PR-00.00.00.016	Laida: A; Data: 5/20/2026; Kalba: lt.; Lapas: 1/1