Ministry of Higher Education and Scientific Research
وزارة التعليم العالي والبحث العلمي

Badji Mokhtar Annaba University
Université Badji Mokhtar – Annaba
Faculty of Technology

Department of Electronics

جامعة باجي مختار – عنابـــة
كـليـة التكنـولـوجيـا
قســم الإلكترونيك

Thesis

Submitted to obtain the diploma of

Doctorate Third Cycle

Field : Electronics

Speciality : Embedded Electronics

By :
HADOUNE Oussama

Title :

Implémentation en temps réel d'algorithmes dédiés au contrôle du mouvement d'un système multiaxes

Thesis defended on 2024/02/08 in front of the jury composed of:

N°	Name and Surname	Grade	Establishment	Quality
01	**Doghmane Noureddine**	Prof.	Badji Mokhtar University – Annaba	President
02	**Benouaret Mohamed**	Prof.	Badji Mokhtar University – Annaba	Supervisor
03	**Arbaoui Fayçal**	Prof.	Badji Mokhtar University – Annaba	Examiner
04	**Saidi Mohamed Larbi**	Prof.	Badji Mokhtar University – Annaba	Examiner
05	**Lachouri Abderrazak**	Prof.	University of 20 August 1995 - Skikda	Examiner
06	**Amara Korba Mohamed Cherif**	MCA	Mohamed Cherif Messaadia University – Souk Ahras	Invited

« تنفيذ في الوقت الحقيقي للخوارزميات المخصصة للتحكم في الحركة لنظام متعدد المحاور»

الملخص:

إن التقدم التكنولوجي والطلب المتزايد على الأنظمة السريعة وعالية الدقة جعل الروبوتات الموازية ذات أهمية متزايدة في العمليات الصناعية والتصنيعية اليوم. على الرغم من توفيرها لمساحة عمل أصغر مقارنة بالروبوتات الأخرى، إلا أن الروبوتات الموازية تقدم حلا موثوقا للتطبيقات التي تتطلب الدقة والاستقرار والتنفيذ عالي السرعة، مما يجعلها أداة حاسمة في الصناعات والأبحاث الحديثة. من ناحية أخرى، لا تزال أنظمة الكرة على اللوحة مهمة في مجال أبحاث وتعليم أنظمة التحكم نظرا لبساطتها وسهولة الوصول إليها وقدرتها على توفير منصة لاختبار وتقييم خوارزميات التحكم. لذلك، يتم استخدام روبوت موازي بـ 3درجات من الحرية للتحكم في الكرة على المستجيب النهائي له.

تهدف هذه الأطروحة إلى تحسين استجابة نظام الاستدلال العصبي الضبابي التكيفي (ANFIS) والذي تم تصميمه وفقا لنظام التغذية الراجعة مزدوج الحلقة. يتطلب نظام الكرة على اللوحة تنفيذ وحدتي تحكم، واحدة لكل محور، مما يزيد من تعقيد الخوارزمية. لحل هذه المشكلة، تقترح الأطروحة تقسيم الخوارزمية على نواتي المتحكم الدقيقESP32. تمت مقارنة أداء خوارزمية ANFIS المنقسمة على نواتي وحدة التحكم مع أدائها في وحدة التحكم أحادية النواة ووحدة التحكم PID التقليدية. وأظهرت النتائج أن الطريقة المقترحة تعمل على تحسين كفاءة الخوارزمية وتقليل زمن الوصول، وهو ما أكدته اختبارات رفض الاضطراب.

كلمات مفتاحية: الوقت الحقيقي، التحكم الضبابي، الشبكات العصبية، التنفيذ، الروبوت الموازي، الكرة على اللوحة، التحكم التكيفي، التصميم بمساعدة الحاسوب.

« Implémentation en temps réel d'algorithmes dédiés au contrôle du mouvement d'un système multiaxes »

Résumé :

Les avancées technologiques et la demande croissante de systèmes rapides et de haute précision ont rendu les robots parallèles de plus en plus pertinents dans les processus industriels et de fabrication d'aujourd'hui. Malgré qu'ils offrent un espace de travail plus petit par rapport à d'autres robots, les robots parallèles présentent une solution fiable et flexible pour les applications qui exigent la précision, la stabilité et l'exécution à grande vitesse, ce qui en fait un outil crucial dans les industries modernes et la recherche. D'un autre côté, les systèmes balle sur plaque continuent d'être importants dans le domaine de la recherche et de l'enseignement des systèmes de contrôle en raison de leur simplicité, de leur accessibilité et de leur capacité à fournir une plateforme pour tester et évaluer les algorithmes de contrôle. Par conséquent, un robot parallèle à 3 Degrés De Liberté (DDL) est utilisé pour contrôler une balle sur son effecteur final.

Cette thèse a pour but d'améliorer la réponse du Système d'Inférence Adaptatif Neuro-Flou (ANFIS), qui a été conçu selon un schéma de rétroaction à double boucle. Le système de la balle sur plaque nécessite l'implémentation de deux contrôleurs, un pour chaque axe, ce qui augmente la complexité de l'algorithme. Pour résoudre ce problème, la thèse propose de diviser l'algorithme en deux cœurs du microcontrôleur ESP32. Les performances de l'algorithme divisé lors de l'utilisation du contrôleur flou ont été comparées à celles d'un monocœur et à un contrôleur PID conventionnel. Les résultats ont montré que la méthode proposée améliore l'efficacité de l'algorithme et réduit la latence, comme confirmé par les tests de rejet de perturbation.

Mots clés : Temps-réel, Contrôle flou, Réseaux de neurones, Implémentation, Robot parallèle, Ball sur plaque, Contrôle adaptatif, Conception assistée par ordinateur.

« Real-time implementation of algorithms dedicated to motion control of a multi-axis system »

Abstract :

Advancements in technology and the growing demand for high precision and fast systems have made parallel robots increasingly relevant in today's industrial and manufacturing processes. Despite that they provide a smaller working space when compared to other robots, parallel robots present a dependable and flexible solution for applications that demand accuracy, stability, and high-speed execution, making them a crucial tool in modern industries and research. On the other hand, the ball-on-plate systems continue to be important in the field of control systems research and education due to their simplicity, accessibility, and ability to provide a platform for testing and evaluating control algorithms. Therefore, a 3-Degree of Freedom (DOF) parallel robot is used to control a ball on its end-effector.

This thesis aims to enhance the control system response of the Adaptive Neuro-Fuzzy Inference System (ANFIS), which was designed with a double-loop feedback scheme. The ball-on-plate system requires two controllers for each axis, leading to a complex algorithm. To tackle this challenge, the thesis suggests dividing the ANFIS implementation into two cores of the ESP32 microcontroller. The performance of the divided algorithm using ANFIS was compared to a single-core implementation and a conventional PID controller. The results indicated that the proposed method improves the algorithm's efficiency and reduces latency, as verified by perturbation rejection tests.

Keywords : Real-time, Fuzzy control, Neural networks, Implementation, Parallel robot, Ball on plate, Adaptive control, Computer-aided design.

To my parents, who have always been a source of inspiration and support in my journey. Their unwavering belief in me has been my driving force, and their love and encouragement have made all the difference.

To my sisters, who have been my rock through thick and thin. Their unwavering love, support, and patience have made this journey easier, and their companionship has been a blessing beyond measure.

To my mentor, who has guided and inspired me throughout my academic journey. His knowledge, wisdom, and experience have been invaluable, and I am grateful for the opportunities he has provided me.

To my friends, who have been a constant source of laughter, support, and encouragement. Their friendship has made this journey more enjoyable, and I am grateful for their companionship.

To all who have supported and believed in me, in ways big and small. Your encouragement and support have meant the world to me, and I could not have done this without you.

ACKNOWLEDGMENTS

First of all, I thank Allah the almighty, for allowing me to reach this modest scientific level and for giving me the courage and the patience to carry out the work done in this thesis.

I am deeply grateful to all those who have supported me during the course of this research project. First and foremost, I would like to express my sincere gratitude to my advisor, Professor **Mohamed Benouaret**, for his invaluable guidance, encouragement, and unwavering support throughout the journey of this thesis.

I would particularly like to thank all the jury members who have accepted to preside and read this work. **Pr.Doghmane Noureddine** from Badji Mokhtar University as jury president, **Pr.Arbaoui Faycel** and **Pr. Saidi Mohamed Larbi** from Badji Mokhtar University Annaba, **Prof. Lachouri Abderrazak** from the University of 20-August 1995 - Skikda, and **Dr. Amara Korba Mohamed Cherif** from Mohamed-Cherif Messaadia University - Souk Ahras. I would also like to thank my colleagues and classmates, **Dr. Aliouat Ahcene**, **Dr. Mansri Islem**, **Zeghida Abdennour** for providing me with a supportive and stimulating academic environment, and **Dr. IDRES Ourida** for her helpful insights and comments.

I am particularly grateful to my family, who have been my source of inspiration and motivation throughout this challenging process. Their love and support have been invaluable and I could not have completed this thesis without them.

Finally, I would like to acknowledge all the individuals who have contributed directly or indirectly to my research. Their contributions have been instrumental in making this project a success.

List of Figures

List of Tables

Acronyms

DC	Direct Current
AC	Alternative Current
RUR	Rossum's Universal Robot
PID	Proportional Integral Derivative
PD	Proportional Derivative
SMC	Sliding Mode Control
FLC	Fuzzy Logic Controller
LQR	Linear Quadratic Regulator
GA	Genetic Algorithm
RL	Reinforcement Learning
DDPG	Deep Deterministic Policy Gradient
MATLAB	Matrix Laboratory
ANFIS	Adaptive Neuro Fuzzy Inference System
DOF	Degree Of Freedom
RPM	Rotation Per Minute
FPGA	Field Programmable Gate Array
PKM	Parallel Kinematic Machine

ETS	Superior Technology School
IFW	Industrial Fire World
SPM	Spatial Parallel Manipulator
TPM	Translational Parallel Manipulator
CKCM	Control of Closed Kinematic Chain Manipulators
NASA	National Aeronautics and Space Administration
VES	Vehicle Emulator System
Idle	Integrated Development and Learning Environment
UPS	Universal Prismatic Spherical
STS	Space Transportation System
VISS	Vibration, Isolation, Suppression and Steering System)
JPL	Jet Propulsion Laboratory
INRIA	National Institute for Research in Digital Science and Technology
ISIS	Intelligent Surgical Instruments & Systems
MBARS	Mini Bone Attached Robotic System
U.S	United States
Fm	Mobile Frame
Fr	Frame of Reference
MPU	Motion Processing Unit
PC	Personal Computer
PET	Polyethylene Terephthalate
MCU	Micro Control Unit
BPS	Ball on Plate System

CAD	Computer Aided Design
CNC	Computer Numerical Control
RAD	Random Access Memory
FPS	Frames Per Second
USB	Universal Serial Bus
SPI	Serial Peripheral Interface
I2C	Inter-Integrated Circuit
UART	Universal Asynchronous Receiver/Transmitter
MOSI	Master Out Slave In
MISO	Master In, Slave Out)
SCK	Serial Clock
SS	Slave Select
ID	Identification
PWM	Pulse Width Modulation
RS485	Recommended Standard 485
EMI	Electromagnetic Interference
Mbps	Megabits Per second
PLCs)	programmable Logic Controllers
HMIs	Human Machine Interfaces
DMIPS	Dhrystone Million Instructions Per Second
WEP	Wired Equivalent Privacy
WPA	Wireless Protected Access
SRAM	Static Random Access Memory
SRAM	Dynamic Random Access Memory

ADC	Analogue to Digital Converter
BLE	Bluetooth Low Energy
IOT	Internet Of Things
NOC	Network On Chip
SOC	System On Chip
CPU	Central Processing Unit
GPU	Graphics Processing Unit
DSPs	Digital Signal Processors
CMOS	Complementary Metal Oxide Semiconductor
BJT	Bipolar Junction Transistor
DAC	Digital To Analog Converter
RMSE	Toot Mean Square Error
HIL	Hardware In the Loop
SIL	Software In the Loop
MB	MegaByte
KB	KiloByte
DXF	Drawing eXchange Format
IC	Integrated Circuit
PMU	Power Management Unit

Contents

Introduction

"Out of clutter, find simplicity. From discord, find harmony. In the middle of difficulty lies Opportunity"

Albert Einstein

Introduction

Context and Motivation

The future of computer and communications systems is reliant on the available hardware. With the rising complexity of artificial intelligence algorithms and the appearance of applications needing significant processing and networking capabilities, such as extended reality, existing chips will become obsolete a few years after they are produced. Indeed, Moore's law predicts that the number of transistors in an integrated circuit will double every two years, leading to improved performance. However, the atomic nature of matter poses a fundamental limit to the evolution of hardware. For instance, the latest Qualcomm Snapdragon 8 Gen 2 processor was manufactured using 4 nm process technology, which approaches the atom's size. As a point of comparison, a silicon atom is about 0.2 nm, and at least two of them are needed with a dielectric layer to construct a transistor [2]. Given these constraints, it is clear that we are rapidly approaching the size limit at which we can no longer manufacture transistors [3]. Consequently, if transistor miniaturization and arrangement reach their limit soon, it is essential to consider alternative solutions to improve chip performance, such as developing programs that take advantage of the use of the existing hardware.

The evolution of mathematics and computer science, including programming languages, libraries, and frameworks, has provided the infrastructure for the development of sophisticated algorithms that need to be executed in real-time. This has led to the creation of new control strategies, including model predictive control [4], intelligent control strategies such as fuzzy logic [5], and neural network-based control systems [6], which gained prominence in recent years, and were used in various applications. Thus, the field of control engineering continues to evolve rapidly, with new control strategies and techniques being developed regularly.

In order to keep pace with this rapid technological advancement, there arises a pressing need for a comprehensive testing platform aimed at evaluating a spectrum of control techniques, particularly for educational purposes. This educational setup should be used as a hands-on learning platform allowing the students to directly interact with the system and observe its behavior, helping them bridge the gap between theoretical knowledge and practical application. Additionally, this platform should ensure safety and incorporate the characteristics of complex real systems, such as non-linearity, multi-variability, and instability. Several benchmarks are available for testing control approaches, including the pendulum on a cart. However, this model has limitations based on the rail's length and could pose damage if the controller is not

well-designed. Similarly, the inverted rotating pendulum is risky and requires strict control. On the other hand, the ball-on-plate system stands out as the most suitable model for testing new control techniques due to its ability to provide a safe environment while incorporating the characteristics of a complex real system and its minimal space requirements compared to other benchmarks.

The ball-on-plate system typically has 4 Degrees-Of-Freedom (DOF) enabling the ball to move freely and remain stable on the plate. However, previous research [7, 8] suggests that the most common ball-on-plate system is based on only 2-DOF, rendering it under-actuated. On the other hand, by the closing half of the 20th century, parallel robots, also known as parallel manipulators, had come into existence. The term parallel is due to their several parallel limbs that support the end effector. This robot is designed to have high precision and increased rigidity, making it ideal for applications that require high accuracy and stability, such as pick and place, packaging, and manufacturing [9]. Thus, this thesis proposes a novel ball-on-plate system based on a 3-DOF parallel robot, a previously unexplored structure. This innovative structure takes advantage of the parallel robot's features to better control the ball on its end effector.

Problem Statement

Different comparison studies between numerous controllers were evaluated on the ball-on-plate system, such as PID and PD [7], PID, Sliding Mode Control (SMC), FLC, and Linear–Quadratic-Regulator (LQR) [10], PID, Genetic Algorithm (GA)-PID, Reinforcement Learning (RL)-PID, deep deterministic policy gradient (DDPG) [11] and between FLC and PID controller [12].

On the one hand, researchers proposed the combination of two controllers to enhance the system response, such as a fuzzy controller to supervise PID parameters [13] and a fuzzy controller combined with SMC to reduce the chattering effect [14]. On the other hand, academic researchers suggested different approaches of control such as Neural-Network based PID controller [15] and fuzzy control with adaptive integral control action [16].

Previous research has focused on proposing new control techniques and approaches for the ball-on-plate system using MATLAB [14, 17–19]. This latter is an excellent software environment for developing, testing and evaluating control algorithms. It provides a wide range of tools and functions for data analysis, visualization, and modeling, making it easy to develop and test complex algorithms. However, while MATLAB is an ideal tool for simulating algorithm development, it is unsuitable for real-time control applications due to its relatively slow processing speed and high resource requirements. On the other hand,

microcontrollers are designed specifically for real-time control applications. They are small, low-power, and efficient, making them ideal for embedded systems.

Therefore, in this thesis, our objective is to implement two control loops on a 32-bit microcontroller. Each control loop incorporates a fuzzy controller with 49 rules. Additionally, the implemented algorithm encompasses an inverse kinematics algorithm that involves numerous mathematical computations, including trigonometric functions and matrix operations. These elements significantly contribute to latency and can potentially decrease the overall performance of the control system and the microcontroller. As a result, our major concern is how to properly implement the control system in the most effective manner by utilizing the microcontroller's maximum available resources to ensure a real-time execution of the proposed control system.

Objectives and Contributions

Based on the cited problems, our objectives focus on proposing a solution that optimizes the utilization of the microcontroller's available resources to improve the system's performance. Since the hardware evolution is limited to the size of atoms, we aim through a comparative analysis to demonstrate the potential for significant performance enhancements through the use of multi-core architectures. Indeed, in our work, the following specific goals are pursued:

- Derive the inverse kinematics of a 3-DOF parallel robot that uses three spherical joints.
- Design of the parallel robot using Computer-Aided Design (CAD) software, followed by manufacturing using a Computer Numerical Control (CNC) machine.
- Design of an Adaptive-Neuro Fuzzy Inference System (ANFIS) controller that is a data-driven technique that can capture the complex relationships present in the data without relying on a prior understanding of the plant's behavior.
- Split the algorithm into two separate cores. This parallel architecture will reduce the computational load on a single core, thereby increasing the overall execution speed and improving the system response.
- Compare the performance of the ANFIS controller with that of a conventional PID controller implemented using single and dual-core configurations.

- Confirm the obtained results through disturbance rejection tests.

In view of the mentioned objectives, we were able to provide the following contributions:

Contribution 1:

J1 Hadoune, Oussama, and Mohamed Benouaret. "ANFIS multi-tasking algorithm implementation scheme for ball-on-plate system stabilization." Indonesian Journal of Electrical Engineering and Informatics (IJEEI) 10.4 (2022): 983-995. https://doi.org/10.52549/ijeei.v10i4.4216

Contribution 2:

C1 Hadoune, Oussama, *et al.*"Tracking control of a ball on plate system using PID controller and Lead-/Lag compensator with a double loop feedback scheme." Avrupa Bilim ve Teknoloji Dergisi 28 (2021): 375-380.

Contribution 3:

C2 Hadoune, Oussama, and Mohamed Benouaret. "Fuzzy-PID tracking control of a ball and plate system using a 6 Degrees-of-Freedom parallel robot." 2022 19th International Multi-Conference on Systems, Signals & Devices (SSD). IEEE, 2022.

Contribution 4:

C3 Hadoune, Oussama, and Mohamed Benouaret. "Control of a 2-DOF parallel robot using Fuzzy Logic Controller." In The first international symposium on industrial engineering maintenance and safety (IEMS), March 05-06, 2022, Oran, ALGERIA 2022.

Papers being prepared for publication:

- Hadoune, Oussama, and Mohamed Benouaret "Artificial Neural-Network based PID controller for the control of the ball-on-plate system"
- Hadoune, Oussama, and Mohamed Benouaret "Real-time implementation of an adaptive system for a bomb defuser mobile robot with a 6-DOF arm"

Thesis outline

The thesis is structured as follows:

Chapter 1 provides an overview of the theoretical background of parallel robots, accompanied by a compre-

hensive review of the literature pertaining to controllers for ball-on-plate systems. Chapter 2 presents the mathematical model of the ball-on-plate system and the 3-DOF parallel robot inverse kinematics. Chapter 3 describes the design of the 3-DOF parallel robot different parts, assembly and manufacturing, including discussions about the hardware configuration and the interaction between the different devices. Moving forward, chapter 4 focuses on the controllers design and real-time implementation of the algorithm using our proposed method to reduce latency. Finally, the last chapter concludes the thesis, highlighting its most important results and contributions. Future works and perspectives are also put forward.

Chapter 1

Background and Literature Review

1.1 Introduction

The Ball-on-Plate System (BPS) is a classical benchmark control problem widely studied in the literature due to its practical applications in robotics, control engineering, and computer vision. In recent years, numerous methods have been proposed to control this system, aiming to achieve excellent stabilizing performance and precise ball position control on the plate. However, despite these advancements, certain gaps and limitations still exist in the current literature, necessitating further research and development.

Therefore, the objective of this chapter is to shed light on the state-of-the-art control strategies for the ball-on-plate system. To lay the foundation for this research, this chapter begins with a thorough analysis of parallel robots, including their components, characteristics, architectures, and applications. Understanding the underlying principles and parallel robots' structures is crucial since the ball-on-plate system is based on a 3-Degrees-Of-Freedom (DOF) parallel robot. To pinpoint the challenges that need to be addressed, this analysis is followed by a detailed literature review of the recent advancements in ball-on-plate system control methods. This in-depth examination allows us to extract valuable insights, ultimately paving the way for further advancements in this field.

1.2 History of Robotics

The writer Karel Čapek in his science fiction play used the term "Robot" for the first time, which denotes the name of a corporation that used robots to serve as slave labor for humanity in 1921. Since then, this term keeps being used in a wide range of man-made things, from hardware to software, that are employed to execute tasks to replace humans [20].

Hundreds of robot definitions are known now, and a simple search of the term "robot" may lead to hundreds of millions of online sites. According to the American Heritage Dictionary, a robot is a mechanical device capable of executing a range of frequently sophisticated human tasks on command or by being programmed in advance [21].

Since the writer Isaac Asimov first used the term "robotics" in 1950, the science of robotics has grown tremendously. He established the famous three laws of robotics in his science fiction book "I, Robot":

(i) A robot may not harm a human being or, through inaction, allow a human to come to harm;

(ii) A robot must obey orders from humans unless doing so would violate the first law;

(iii) A robot must defend its own existence so long as doing so does not violate the first or second

law [22].

It is noticeable that all the robots have one thing in common: they can move and be programmed to perform physical tasks. The robot's mechanical structure defines its application field and differs from one robot to another. Therefore, they take on different forms, from humanoid, which imitates the human in form and movement, to industrial robots whose appearance is determined by the task they are intended to perform [23]. The structure of robots could be generally classified as mobile and manipulator, and a manipulator robot could be classified as serial and parallel. As its name indicates, a serial robot is made of connected links in a serial manner. Despite having a large workspace, this type of robot has several weaknesses that lead researchers and engineers to search for another alternative.

1.3 History of Parallel Robots

The need for industrial robots to operate at high levels is perpetually growing. Higher standards for operational precision, load capacity, task flexibility, reliability and speed execution have resulted from the necessity for a completely developed industry. Higher precision assembly, quicker product handling, better measurements, surface finishing, and milling capabilities are some of such needs. Using parallel robots with excellent potential capabilities, such as high stiffness, high precision, and high loading capacities is a common trend to meet these expectations and criteria.

Mathematicians in England and France developed a serious interest in polyhedra centuries ago. The first theoretical research concerning parallel mechanisms, notably six-strut platforms, was born out of this fascination. However, a relatively small number of academics read and studied these publications [24].

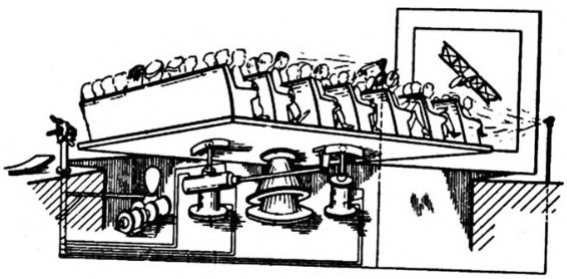


Figure 1.1: The first spatial parallel robot, patented in 1931 [25]

.

In 1928, James E. Gwinnet created a conceptual amusement device based on a parallel robot as depicted

in Figure 1.1. At that time, the entertainment industry did not pay attention to such a discovery [26]. Ten years later, Willard L.V. Pollard designed a parallel robot for automatic spray painting, which was claimed as the first parallel robot. However, this robot was never actually built. Conversely, Pollard's son, Willard L.V. Pollard Jr., conceived and built the first industrial parallel robot shown in Figure 1.2

[24].

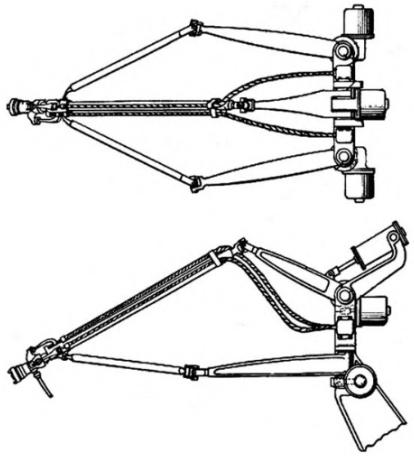
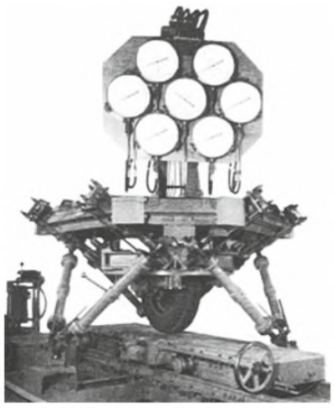


Figure 1.2: The first spatial industrial parallel robot, patented in 1942 [24]

.

The universal rig, the first octahedral hexapod with six degrees of freedom that Dr. Eric Gough created, revolutionized the robotics industry in 1947. It served as a tire-testing tool to gather data regarding the characteristics of tires that were subjected to varied loads, as seen in Figure 1.3a. Figure 1.3b shows a tire-testing device that was in use from 1954 until it was decommissioned in 2000 [24].



(a) The first Gough Stewart platform

(b) The last Gough platform tire-testing machine

Figure 1.3: The tire-testing machine of Gough Stewart [24]

In 1967, Klaus Cappel who is an engineer at the Franklin Institute Research Laboratories in Philadelphia, received a patent for inventing a flight simulator, as is depicted in Figure 1.4. Cappel is considered one of the last pioneers in the area of parallel robotics, following in the footsteps of Eric Gough and Dr. Stewart [24].

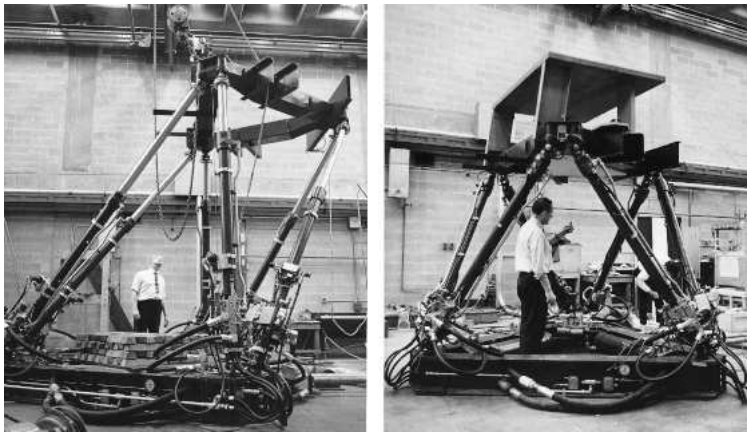


Figure 1.4: Cappel's Stewart platform's aerial flight simulator [27]

.

As we mentioned earlier, the parallel robot's application field is determined by the shape of its end-effector. Thus, in this thesis, the chosen parallel robot's end-effector is a plate that allows for the free-rolling of a ball, with the objective of balancing it. Further information about this mechatronics system will be discussed in the next section.

1.4 Basic Components of Parallel Robots

A parallel robot is a complex system that can incorporate various components from different fields, such as electrical, mechanical, hydraulic, and pneumatic components. Therefore, controlling a parallel robot requires a solid understanding of the basic operating principles of the components used in the robot. Table 1.1 shows the different parts that can be used to construct a parallel robot, including the body, joints, and end-effector.

functional blocks	Components
Structure	Articulated mechanical structure
Sensor devices	Exteroceptive sensors
	Proprioceptive sensors
Motor devices	Actuators: Electrical
	Pneumatic
	Hydraulic
Control system	Control strategy
Processor	Hardware
	Embedded algorithm

Table 1.1: Basic components of a parallel robot

1.4.1 Body and Joints

The body of a parallel robot consists of fixed and mobile parts connected through links. These links provide structural and physical properties and are attached using joints. The type of joint constrains the parallel robot's movements, thereby determining its motion and application field. There are various types of joints, including the slider joint, which allows for translation along one axis and is widely used in parallel robots like prismatic actuators. Prismatic actuators are well known for their high level of precision and velocity in a wide range of applications. The ball joint is another type of joint found in parallel robots. It allows for rotation along the x, y, and z axes, enabling objects connected through ball joints to move up and down, left and right, and backward and forward.

When describing the orientation of a moving object in space, pitch, roll, and yaw angles are defined as rotations along the lateral, longitudinal, and vertical axes, respectively. Figure 1.5 depicts these angles on an airplane. The pitch angle denotes the airplane's orientation about the lateral axis, the roll angle indicates the airplane's rotation about the longitudinal axis, and the yaw angle marks the airplane's rotation about the vertical axis [28].

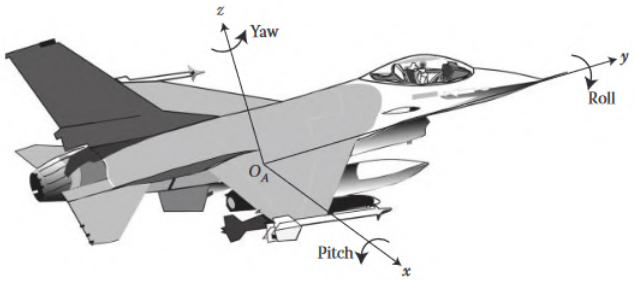


Figure 1.5: Definition of pitch, roll and yaw angles on an airplane [28]

1.4.2 End-effector

The end-effector is the component attached to the robot's end, which is used to accomplish the task that the robot is built for, allowing the robot to interact with the environment. It can be a gripper or process tool such as a spindle or a sensor. Therefore, the end-effector determines the robot's task. Some industrial parallel robot milling machines have recently been equipped with a new feature called a 'tool changer'. This feature enables the robot to auto-change its end-effector according to programmed software, increasing its versatility and production speed.

1.4.3 Actuator

Robots generally use electrical motors because of their low cost and ease of use. Moreover, precision and repeatability are relatively better when using servo motors compared to hydraulic systems. However, industrial robots use hydraulic or pneumatic actuators that provide more power and are more resistant to the aggressive industrial environment. In parallel robots, we can find both electrical and hydraulic types of actuators, but hydraulic actuators are the most commonly used, especially in pick-and-place applications with delta robots. Similarly, simulators using the Stewart platform rely heavily on hydraulic actuators.

1.4.4 Sensors

In robotics, the sensor is an important component [29]. There are two main types of sensors: exteroceptive and proprioceptive. Exteroceptive sensors provide information about the robot's environment, such as infrared sensors that can be used to measure the distance between the robot and nearby objects. It is composed of a transmitter and a receiver. The infrared transmitter sends a laser ray with a constant speed

of 300 million meters per second. Once an object reflects it, it comes back to the receiver. The time between sending and receiving the signal is calculated and that's how we can determine the distance between the robot and the object.

The second class of sensors is known as a proprioceptive sensor. This type of sensor provides information about the robot's internal state, such as an encoder used to measure an electrical motor's Rotation Per Minute (RPM). A light-based encoder comprises a wheel with holes that block and unblock the light, as illustrated in Figure 1.6. This information can be exploited to determine the distance the robot drives or the speed of the motor.

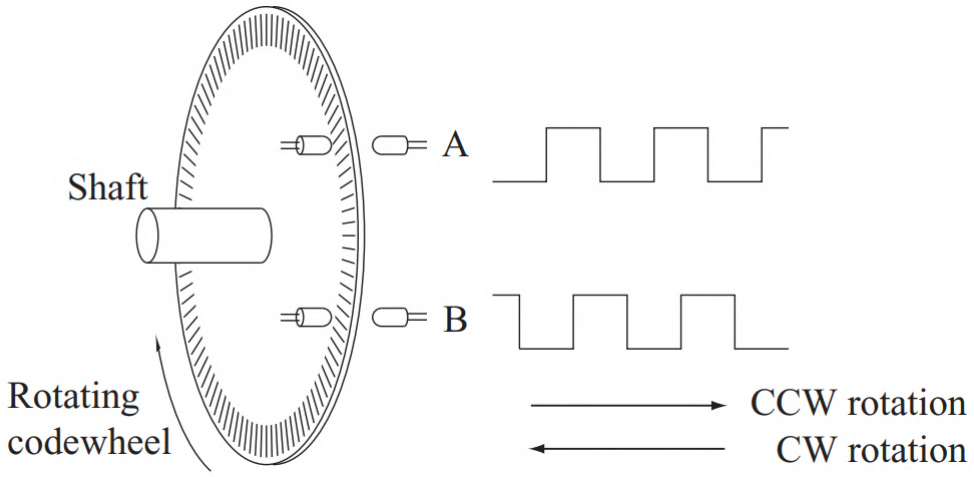


Figure 1.6: An example of proprioceptive sensor [30]

.

1.4.5 Control System

The control system is the software that manipulates the robot's motion. It receives information from the proprioceptive and exteroceptive sensors. According to the received data, the control system decides whether the robot should move forward, backward, or stop. If the robot reaches its desired position, it proceeds to the next task. The robot should move backward and correct itself based on the designed controller if it exceeds its desired position. If the robot doesn't reach the desired position, the control system should provide adequate effort to the actuators to achieve the set point. In some applications where the velocity, torque, acceleration, and deceleration controls are necessary, the controller should take them into account [31].

1.4.6 Processor

The processor processes the received information through the input port and calculates the provided effort, which is then sent to the actuators via the output port. The processor can be considered the brain of the robot. The processor calculates each joint's angles, velocity, acceleration, deceleration and the torque that should be applied to the manipulated object. A large variety of processors differ by the internal architecture, the lithography, which is the least distance between transistors of a processor (the lower the lithography, the faster and more power efficient the processor is) and the clock speed. For applications that consume many resources, processors based on the Intel® Atom processor family & FPGAs are the most suitable.

1.5 Robot Characteristics

We delve into the specifications of the robots, including their payload capacity, workspace dimensions, and repeatability that are defined in the following subsections.

1.5.1 Payload

The payload is the weight a robot can support without changing its specifications. In other words, the payload is the weight that a robot can carry while keeping the same accuracy, speed, and repeatability. On the other hand, the robot's maximum load capacity is the robot's ability to support loads regardless of the changes that may occur on the robot's specifications, such as excessive deflections and slowness in task execution. Parallel robots are known for their capacity to lift heavy weights because of their architecture which allows the weight to be shared on the different parallel robot legs [24].

1.5.2 Workspace

The robot workspace is the maximum distance limit the robot's end-effector can reach. The workspace depends on the robot's joints, legs' length and overall architecture. This specification is important for industrial applications and must be considered before choosing the robot's type. Serial robots are known for their wide range of workspace compared to parallel robots.

1.5.3 Precision

The precision is the difference between the desired position and the reached position. It depends on many parameters, such as the actuators' resolution, sensors' tolerance, measurement accuracy and the embedded control system. The precision can be determined by testing the robot's position and orientation using different values for speed and load-carrying capacity.

1.5.4 Repeatability

The capacity to accomplish repetitive tasks while maintaining the same level of accuracy is known as repeatability. For instance, if a robot is designed to reach a desired point 100 times, it may miss the set point due to various factors. However, it will be within a certain radius of the desired point [32]. This radius is known as repeatability, as illustrated in Figure 1.7.

Repeatability is more crucial than precision because if a robot has a consistent error, it can be predicted, determined, and corrected. Consider a robot that consistently errs 0.5 cm to the left. This error can be easily corrected by adjusting all the desired points 0.5 cm to the right, thus eliminating the error [32]. In contrast, repeatability indicates a random error that cannot be predicted or eliminated.

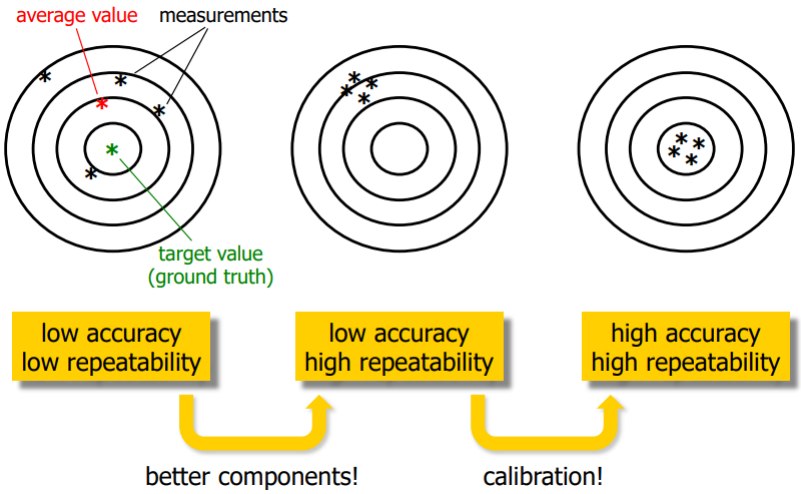


Figure 1.7: The difference between repeatability and precision [33]

.

1.6 Parallel Robots Architectures

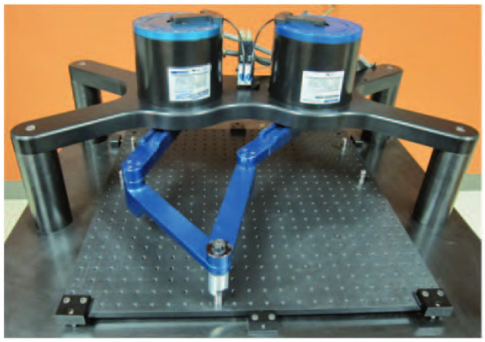
The number of typologies of parallel robots is so vast that it is impossible to list them all here. For example, if we consider a parallel robot with six legs, and all the legs are made of six joints, the number of architectures for such a robot is equal to the number of serial legs raised to the power of six [34]. This number is too large to explore fully in a single thesis. Therefore, we will categorize parallel robots based on their degrees of freedom.

1.6.1 Planar Parallel Robot

As their name implies, planar parallel robots are made for movement in a single plane. They are often referred to as planar parallel manipulators and can be grouped into three main categories:

1. Robots with 2 DOF able to position a point in a plane as presented in Figure 1.8;
2. Robots with 2 DOF that can position a device with constant orientation in a plane as illustrated in Figure 1.9;
3. Robots with 3 DOF able to position a device in a plane as shown in Figure 1.10 [34].

(a) The planar 5-bar mechanism [35]

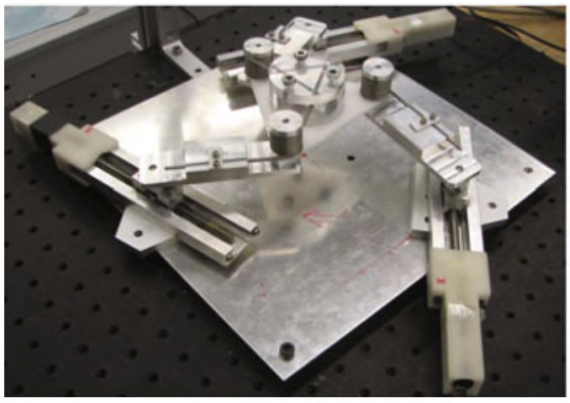


(b) The paraplacer robot [36]

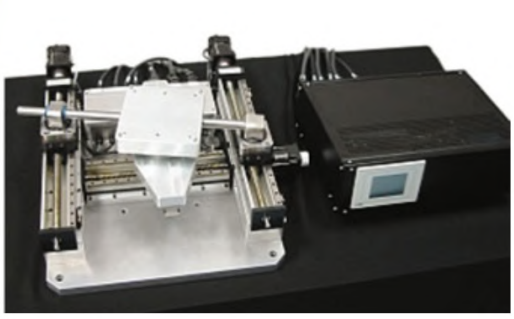
Figure 1.8: A 2-DOF planar robot

Figure 1.9: Pacdrive D2 robot by Schneider [37]

.



(a) Prototype of a 3-Prismatic-Revolute-Revolute (PRR) robot



(b) A decoupled planar robot

Figure 1.10: Examples of robots with 3-DOF [34]

The planar parallel robots are among the most popular and widely used in various industries. Another type, known as spatial parallel robots, will be covered in the next section.

1.6.2 Spatial Parallel Manipulator

Spatial Parallel Manipulators (SPM), have been designed to move their platform in space. This type of parallel robot has various and numerous categories, some of which are listed below:

1. Robots with three translational DOF are called Translational Parallel Manipulators (TPM)): among them, we can mention the Delta robot in Figure 1.11a, the Orthoglide presented in Figure 1.11b) as well as the Tripteron illustrated in Figure 1.11c).

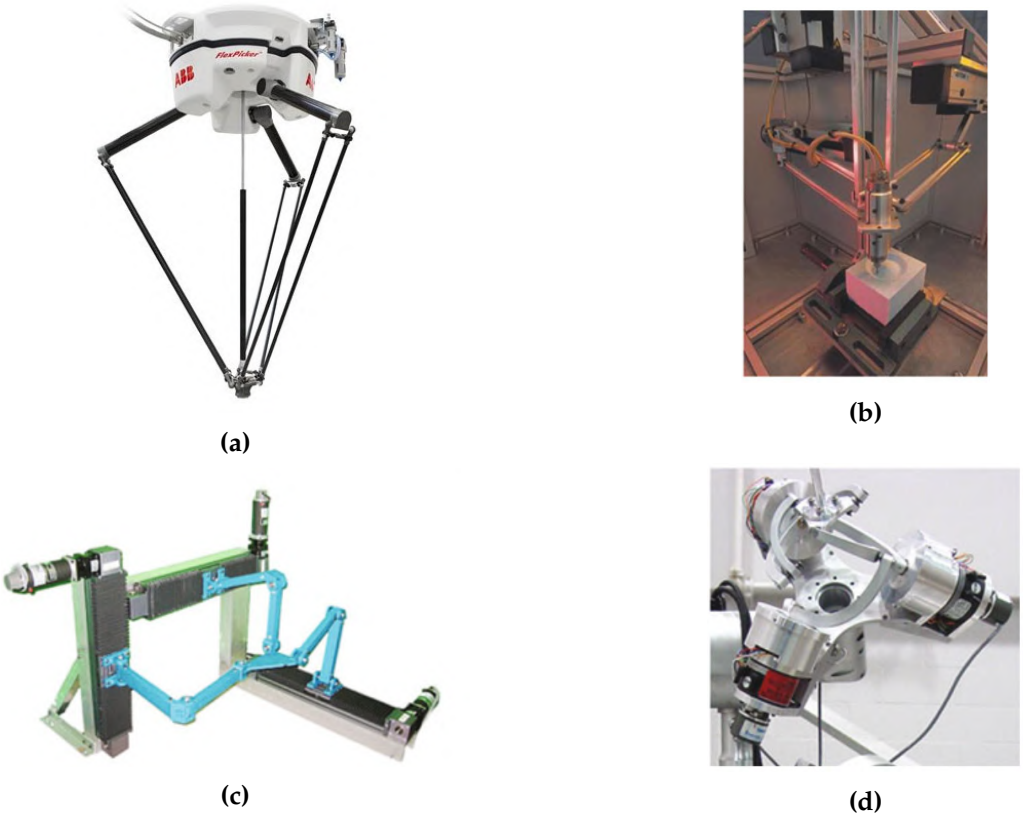


Figure 1.11: Examples of spatial robots with 3 translational or rotational DOF [38]

2. Robots with three rotational DOF are called spherical parallel kinematic machines: One of their most common features is that the platform can rotate around a fixed point. The most-known is the Agile Eye, displayed in Figure 1.11d.

3. Robots with three exotic degrees of freedom: These sorts of robots often have some DOF of rotation restricted by the DOF of translation. Some of them were designed with an extra wrist to counteract the undesired rotations, and they have found some industrial uses, particularly for milling, as illustrated in Figure 1.12.

Figure 1.12: Spatial robots with 3 exotic DOF [34]

.

4. Robots with three translational DOF and one rotational DOF around one given axis are called Schönflies motion generators. Pick-and-place operations are typically carried out with them, often at high speeds. The adept Quattro is one of their most known robots, shown in Figure 1.13.

Figure 1.13: The pick and place Quattro parallel robot [39]

.

5. The Gough-Stewart platform is a six-DOF parallel robot presented in Figure 1.14 [34].

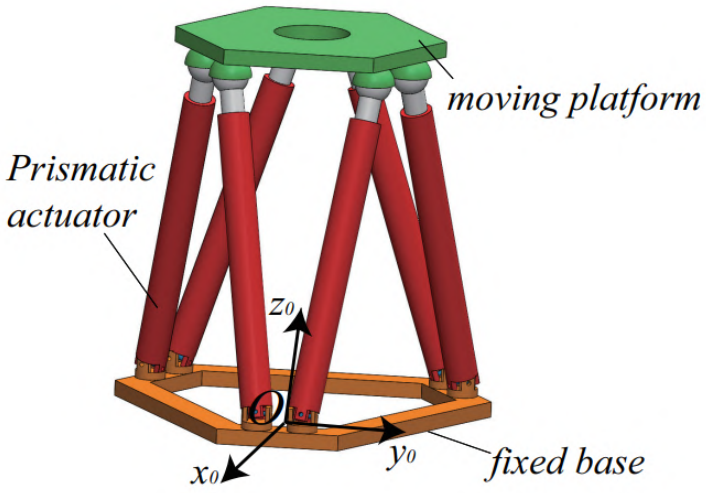


Figure 1.14: Example of 6-DOF spatial parallel robot [40]

.

1.6.3 Hybrid Robot

The hybrid robot, as depicted in Figure 1.15, consists of parallel modules that are combined serially. This serial arrangement of the hybrid manipulators results in increased overall stiffness and improved response qualities, thereby addressing the limitations of workspace constraint in parallel manipulators [34].

Figure 1.15: Example of a Serial-Parallel Hybrid Leg Mechanism [41]

1.6.4 Cable-Driven Parallel Robot

One relatively recent type of parallel kinematic machine uses cables instead of rigid connections for motion control. To properly control a 6-DOF platform, at least seven cables are needed as they are only utilized in tension [42]. The Skycam is one of the most recognized examples of a cable-driven parallel robot depicted in Figure 1.16.

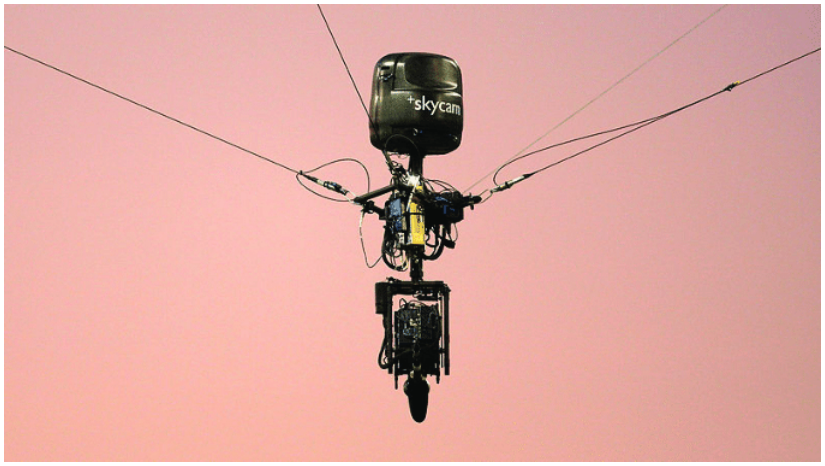


Figure 1.16: Example of Skycam cable driven parallel robot [43]

.

These parallel manipulators outperform conventional parallel robots in many ways. They have light construction with low inertia of movement and large working space, decreasing the mass of the mobile platform and making them particularly appropriate for high-speed and high-performance applications. Other features include a high payload-to-weight ratio, speed, and mobility. The mechanical system has a basic construction and is inexpensive to build. Another significant benefit is that they are re-configurable, allowing them to be utilized for different tasks by simply adjusting the cable attachment points. The primary disadvantage of cable-driven manipulators is related to the physical properties of cables. Because cables can only pull but not push, they must be kept tight while the manipulator is in operation, which complicates their control [42].

1.7 Application of Parallel Robots

The use of parallel structures is widespread and encompasses many applications, making it difficult to mention them all. Therefore, we select a few representative examples to be presented in the subsequent

subsections.

1.7.1 Spatial Applications

Parallel mechanisms have been a topic of interest in developing spatial devices for a considerable time. One of the earliest applications was seen in the landing gear of the lunar module. Truss structures are ideal for creating reconfigurable spatial structures, while inflatable hexapods have been considered to deploy lightweight structures in space. McDonnell Douglas (now Boeing) designed a tendon-suspended platform robot named Charlotte to automate crew tasks, and it was used in the STS-63 space shuttle mission in February 1995, depicted in Figure 1.17 [9, 44].

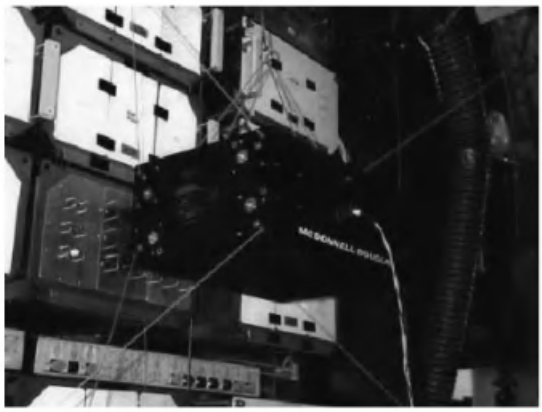


Figure 1.17: The Charlotte's tendon suspended platform robot designed by McDonnell Douglas that has been tested in the space shuttle mission STS-63 [9]

.

Using parallel structures in telescopes is highly effective and has become a common practice. Hexapods are widely used in current land-based telescopes, serving various purposes such as aligning secondary mirrors, pointing the primary mirror, or being used as a research instrument, as depicted in Figure 1.18. These hexapods have proven to be a reliable and efficient solution for telescopes, providing the necessary precision and stability required for astronomical observations [9].

Figure 1.18: Hexapod used for secondary mirror alignment [9]

.

1.7.2 Vibration

Parallel structures have been considered attractive options for vibration dampening due to their ability to provide large bandwidth. An interesting example of this is the Vibration, Isolation, Suppression, and Steering System (VISS), which was developed by the American Air Force, Honeywell, Trisys and JPL (Jet Propulsion Laboratory). The VISS was designed to isolate onboard measuring instruments, such as optical and laser devices, from the body of a satellite. This system has been successfully used on various satellites, as shown in Figure 1.19 [38, 45].

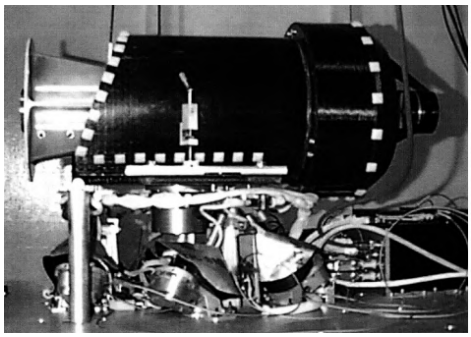


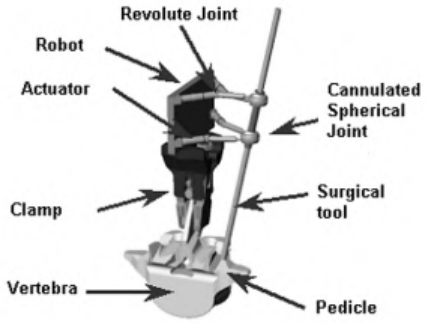
Figure 1.19: VISS hexapod configuration [38]

1.7.3 Medical Applications

The medical field is gradually infusing robots like the Da Vinci and the Zeus robot [9]. Parallel structures have been considered for their potential use in the medical field early on, as demonstrated by the Crigos system by Brandt [46], which utilized a parallel robot for orthopedic surgery. In addition, the active wrist of the National Institute for Research in Digital Science and Technology (INRIA) was successfully used in eye surgery on dogs [47]. Contrary, only a few of the laboratory prototypes of serial robots have found practical applications, but there are a couple of exceptions. One example of the parallel structure is the SurgiScope system by the Intelligent Surgical Instruments Systems (ISIS) robotics, which employs a Delta-type robot as a microscope stand, as depicted in Figure 1.20a.

Another challenge in surgical robotics is the ability to follow the patient's movements. This has led to the development of the MiniAture Robot for Surgical procedures (MARS), which has a 6- Universal Prismatic Spherical (UPS) structure [48]. It is directly mounted on the patient's bones near the surgical site. The MARS has been utilized as a surgical tool for guiding the placement of spinal pedicle screws and is marketed as the spine assist robot by Mazor. A similar robot for knee arthroplasty is known as the MBARS (Mini Bone-Attached Robotic System). It was developed at the Robotics Institute Carnegie Mellon University (CMU) and is depicted in Figure 1.20b. The spine assist and MBARS suggested a trend towards smaller, adaptable, and cost-effective surgical robots, compared to the expensive and large-scale structures [9].

(a) Using Delta robot as a microscope stand by the ISIS system [40]



(b) Mini Bone-Attached Robotic System [49]

Figure 1.20: Example of medical applications of parallel robots

Additionally, parallel structures are far less susceptible to the scaling effect than their serial equivalent, making them suitable for micro-robots. This type of robot is ideal for minimally invasive surgery in medical applications, especially endoscopy [9].

1.7.4 Simulators

Over the years, several parallel robots have been created for flight simulators, starting with Stewart's original idea. An increasing number of companies are designing virtual reality motion simulators for airplanes, boats, trains, and trucks. Parallel structures are likely to be most successful in this sector. Figure 1.21 displays a remarkable parallel robot built with hydraulic actuators by the US Army Center for Tanks Research. It can carry 27 tons and withstand vertical accelerations from 4 to 6 g. This tank aims to test the ergonomics of the tank's interior, and the arm stabilization [9].

Figure 1.21: The turret motion-based simulator of the US army [9]

.

1.7.5 Industrial Applications

Parallel robots have a wide range of industrial applications, including assembly, material handling, packaging, and machining. Due to their high accuracy, speed, and payload capacity, parallel robots are often preferred over serial robots in tasks that require precision and rapid motion. In the following subsections, we will present some of the different industrial applications of parallel robots.

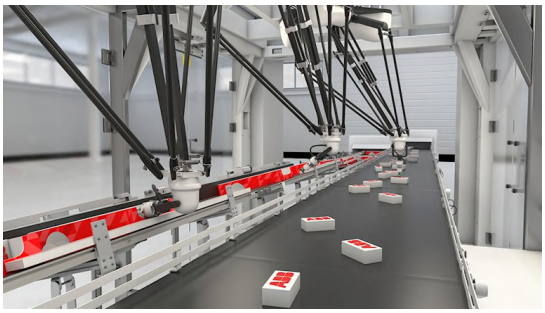
1.7.5.1 Parallel Robot-based Machine Tools

A parallel robot-based machine tool is called a parallel kinematic machine. As compared to conventional machine tools, this technology provides higher stiffness and precision, as well as prospective benefits, such as high dexterity and multi-mode manufacturing. Giddings and Lewis compared the Hexapod machine tool to normal machine tools and discovered that the Hexapod machine tool was much more precise (about seven times), stiffer (roughly five times) and quicker (about four times) [24]. Figure 1.22a depicts the first milling machine based on the Stewart platform principle.

1.7.5.2 Pick and Place Parallel Robot

Pick-and-place refers to a particular type of movement used in modern industry. More specifically, it designates the operation that allows a part to be moved from one location to another, as depicted in Figure 1.22b. Parallel robots are particularly specialized in performing pick-place operations. The automation of this procedure aids in increasing production rates and allowing human workers to concentrate on more difficult duties [50].

(a) The first milling machine based on the principle of Stewart platform [24]



(b) Pick and place parallel robot application [51]

Figure 1.22: Example of industrial applications of parallel robots

1.8 Kinematics of Parallel Robots

The kinematics of parallel robots refers to studying their movement and position relative to a fixed reference frame. Parallel robots are unique because they consist of multiple kinematic chains, which work in parallel to perform a task. The kinematics of parallel robots is an important aspect affecting their accuracy and overall performance. The parallel robot kinematic analysis aims to determine the relationship between the end-effector and the different joints. This relationship is mathematically described using kinematic equations, which depict the motion and behavior of the robot. Thus, the kinematics of parallel robots plays a critical role in their functionality and performance, necessitating a thorough understanding for an effective operation.

The purpose of studying inverse kinematics is to determine the angles of the robot's joints required to achieve a desired end-effector position or orientation. Therefore, in this section, we will explore ways to represent objects, positions, orientations and movements.

1.8.1 Position and Orientation of a Rigid Body

1.8.1.1 Position & Rotation Matrix

To locate any point in three-dimensional space, a set of coordinates must be assigned to that point. Once a reference coordinate system is established, these coordinates can be represented as a (3×1) vector. An

example of a reference coordinate system is the orthogonal coordinate system (O_a, $x_a y_a z_a$), which consists of mutually perpendicular axes and an origin at O_a. Thus, any point p in the space can be represented mathematically using the orthogonal coordinate system as follows:

$$^{a}\mathrm{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

where p_x, p_y, and p_z stand for the vector's components along axes x_a, y_a, and z_a respectively. As seen in Figure 1.23, p is referred to as the position vector in this context.

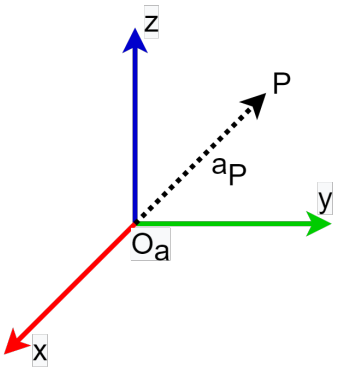


Figure 1.23: Representation of a point in the space

To analyze the motion and manipulation of robots, it is necessary to describe the position and orientation of the robot. Suppose that point b has an orthogonal coordinate system (O_b, $x_b y_b z_b$) associated with it to specify its orientation. Here, the unit vectors of the coordinate axes are denoted as x_b, y_b and z_b. As follows, we express the orientation of point b with respect to the coordinate system (O_a, $x_a y_a z_a$):

$$^{a}_{b}\mathbf{R} = {}^{a}\mathrm{x}_b\, {}^{a}\mathrm{y}_b\, {}^{a}\mathrm{z}_b = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{1.1}$$

$^{a}_{b}\mathbf{R}$ is a rotation matrix that consists of nine elements. It is possible to compute the rotation matrix with respect to the rotation transformation by an angle θ about the axes x, y, and z, respectively [24], as follows:

$$Rx = \begin{bmatrix} 1 & 0 & 0, \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \tag{1.2}$$

$$Ry = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{1.3}$$

$$Rz = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

Assume that the coordinate frames {B} and {A} have the same orientation. However, both coordinate frames' original points are not the same. ${}^{a}\mathrm{P}_{O_b}$ is the translational vector of frame {B} with respect to the frame {A}. We express the position vector of p with respect to the frame {A} as follows:

$${}^{a}\mathrm{P} = {}^{b}\mathrm{P} + {}^{a}\mathrm{P}_{O'_b} \tag{1.5}$$

This is called the translation equation and is presented in Figure 1.24

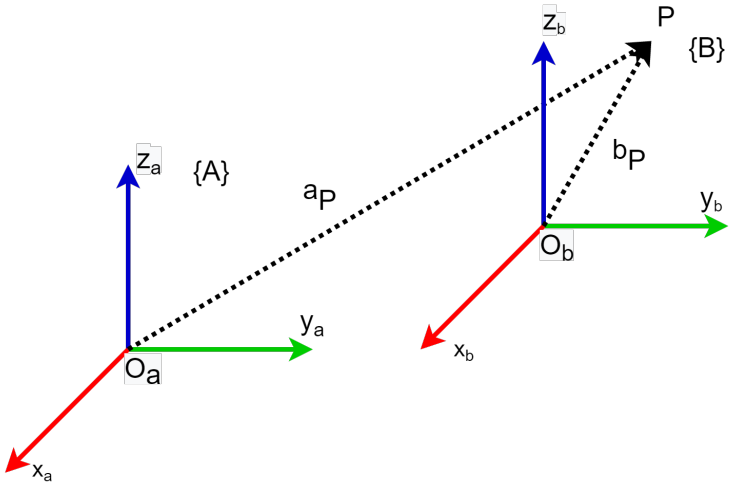


Figure 1.24: Frames' translation

Let's assume that the coordinate frames {B} and {A} share the same origin but have different orienta-

tions. The rotation matrix ${}^a_b\mathbf{R}$ specifies the orientation of the frame {B} with respect to the frame {A}. Hence, we can express the transformation of point p in the frame {A} as follows:

$$ {}^aP = {}^a_b\mathbf{R}.{}^bP \tag{1.6}$$

Equation 1.6 is called the equation of coordinate rotation and is shown in Figure 1.25 where ${}^aP$ indicates the position p with the reference coordinate system {A}, and ${}^bP$ denotes the position p with the reference coordinate system {B}.

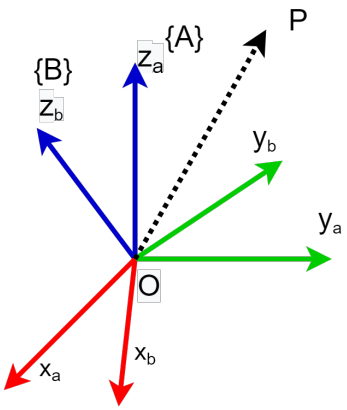


Figure 1.25: Rotational transformation

Based on the common scenario, there is no overlap or similarity in the orientation between the initial points of the frames {A} and {B}. To establish the transformation between them, the position vector ${}^aP_{O_b}$ is employed to denote the original point of frame {B} with reference to frame {A}. Additionally, the rotation matrix ${}^b_a\mathbf{R}$ is utilized to indicate the orientation of frame {B} with reference to frame {A}, as described in [24]. Hence, the transformation can be obtained as follows:

$$ {}^aP = {}^a_b\mathbf{R}.{}^bP + {}^aP_{O_b} \tag{1.7}$$

1.8.1.2 Euler Angles

The Euler angles define the rotation of a rigid body with reference to a fixed frame {Fr}, in a three-dimensional space. The Euler angles are determined by applying three successive rotations around three perpendicular axes. The choice of rotation angles is not unique and depends on the problem to be studied. Rotation angles ϕ, θ and γ around the new x-axis, y-axis and z-axis are defined as follows:

$$Rx = \begin{bmatrix} 1 & 0 & 0, \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{1.8}$$

$$Ry = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{1.9}$$

$$Rz = \begin{bmatrix} cos(\gamma) & sin(\gamma) & 0 \\ -sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.10}$$

The matrix representing this rotation can be expressed as the product of three matrices, each of which describes a rotation around a single axis. Precisely, these matrices are given by:

$$R = Rz(\phi)Ru'(\theta)Rw'(\gamma)$$

$$= \begin{bmatrix} c(\phi)c(\gamma) - s(\phi)s(\gamma)c(\theta) & -c(\phi)s(\gamma) - (\phi)c(\gamma)c(\theta) & s(\gamma)s(\theta) \\ s(\phi)c(\gamma) + c(\phi)s(\gamma)c(\theta) & -(\phi)s(\gamma) + c(\phi)c(\gamma)c(\theta) & -c(\phi)s(\theta) \\ s(\gamma)s(\theta) & c(\theta)s(\theta) & c(\theta) \end{bmatrix} \tag{1.11}$$

where c=cos and s=sin.

1.8.2 Homogeneous Transformation

If the coordinates of any point in an orthogonal coordinate system are known, homogeneous transformation can be used to calculate the coordinates of that point in another orthogonal coordinate system. The homogeneous transformation can be rewritten as follows:

$${}^{a}_{b}\mathbf{T} = \begin{bmatrix} {}^{a}_{b}\mathbf{R} & {}^{a}\mathbf{P}_{O_b} \\ 0_{4X1} & 1 \end{bmatrix} \tag{1.12}$$

We express the position vector of p with respect to the frame A as follows:

$${}^{a}\mathrm{P} = {}^{a}_{b}\mathbf{T}.{}^{b}\mathrm{P} \tag{1.13}$$

Given a point in space described by the vector ai+bj+ck, where i, j, and k are the unit vector of the axes x, y, and z, respectively. The translational homogeneous transformation matrix presented in 1.14 can be employed to represent this point.

$$\text{Trans(a,b,c)} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.14}$$

where Trans refers to the translational transformation, the blue and red circles indicate the translational vector and the diagonal of the rotation matrix, respectively.

If a rigid body rotates about the x, y and z-axis by an angle Θ, then the following equations can be derived:

$$Rot(x,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\theta & -s\theta & 0 \\ 0 & s\theta & c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.15}$$

$$Rot(y,\theta) = \begin{bmatrix} c\theta & 0 & s\theta & 0 \\ 0 & 1 & 0 & 0 \\ -s\theta & 0 & c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.16}$$

$$Rot(z,\theta) = \begin{bmatrix} c\theta & -s\theta & 0 & 0 \\ s\theta & c\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.17}$$

where Rot stands for rotational transformation.

1.9 Review of Existing Studies

Besides its potential for use in various fields, the ball-on-plate system is widely used for reasons related to its distinctive features, such as non-linearity, instability, and uncertainty. These characteristics allow it to be a test bed for researchers to evaluate new control algorithms. This interest has been explained by the publication of several research papers exploring different control techniques for stabilizing the system.

1.9.1 PID Controller

Proportional-Derivative-Integral (PID) controller is the most widely used control system for controlling the BPS. However, achieving satisfactory results with this conventional controller takes work. For example, the author in [52] demonstrated that the system response for stabilizing the ball using a conventional PID controller suffers from a long settling time and a high rate of overshoot. In [53], the authors carried out a comparison between PID, Proportional-Derivative (PD) and Proportional-Sum-Derivative (PSD) controllers for tracking trajectory. The results showed that the PD controller's performance demonstrated a smaller overshoot when compared to the other controllers. To minimize the integral square error, [54] conducted a comparative study between Integer-Order-Proportional-Derivative (IOPD) and Fractional-Order-Proportional-Derivative (FOPD) for tracking trajectory purposes using three different algorithms. The outcome illustrated that when using the same tuned algorithm, the FOPD outperforms the IOPD. Recent studies have also explored the use of PID controllers. [7], for instance, presented a comparison between PID and PD controllers that were applied to a designed and constructed 2-DOF BPS. The ball's position is tracked using a USB HD camera that uses the OpenCV software library. The lighting conditions required for the system to function properly made it difficult for the conventional PID controller to perform well. As a result, the PD algorithm was found to be a more effective solution.

1.9.2 Linear Quadratic Controller

Like the PID controller, the Linear Quadratic Controller (LQR) is considered a feedback controller. Nevertheless, instead of calculating the control signal based on the error between the desired and actual values, the LQR computes the control signal by modeling it as an optimization problem. This technique was used by Kassem *et al.* [10] and Cheng *et al.* [55], where Kassem *et al.*proposed four control strategies: PID, Sliding Mode Control (SMC), Fuzzy Logic Controller (FLC) and LQR. According to the experimental results,

the SMC control technique proved to be more effective than the LQR and the other controllers. On the other hand, Cheng *et al.*implemented the LQR technique on a real system to stabilize the ball at the central point of the plate. The results demonstrated a steady-state error of 7 mm and 15 mm in static and dynamic tracking, respectively.

1.9.3 State Feedback Controller

A state feedback controller was simulated in MATLAB by [56] and [57]. On the one hand, Núñez *et al.*linearized the BPS dynamics around the operating point before the controller design. When the ball was moved by 10 mm, the outcomes showed an accuracy of 3 mm for the ball stabilization with 1 s of settling time. On the other hand, Huang *et al.*utilized the input-output linearization approach to design the stabilization control law. The proposed nonlinear control law proved to be capable of regulating the states of the system to zero asymptotically.

1.9.4 Back-stepping Controller

Ma *et al.* [17] designed an observer-integrated back-stepping control. The authors used a linear extended state observer and a Tracking Differentiator (TD) to estimate the uncertainties of the model and the virtual control derivatives in the back-stepping design. Based on the experiments with circular and square trajectories, the suggested control approach can mitigate and decouple uncertainties while also achieving good tracking performance. On the other hand, Zhankui *et al.* [58] presented an adaptive backstepping sliding mode control with fuzzy monitoring strategy, with the aim of softening the control signal. The experimental results demonstrated the superior performance of the proposed technique in the presence of external disturbances.

1.9.5 Fuzzy Logic-Based Controller

Yubazaki *et al.*designed an FLC in [59] based on a Single Input Rule Module (SIRM). The designed controller showed a good tracking performance with an error of less than 50 mm. In addition, Wang *et al.* [60] proposed a control structure composed of a double-loop feedback scheme, where the inner loop aims to control the angle of the servo motor, the outer loop controls the ball's position. The authors incorporated a fuzzy logic controller with three inputs: position, velocity and acceleration. Similarly, Bai *et al.* [61] de-

veloped a supervisory FLC composed of two layers: the lower defined a fuzzy SIRM. The second layer performed the supervisory task for the ball's velocity and acceleration when following a specific trajectory. However, the findings showed a steady state error in the system's output. Furthermore, Pattanapong and Deelertpaiboon [62] designed a fuzzy PD controller with an adaptive integral action to eliminate the steady state error in static and dynamic tracking. Nevertheless, the outcomes indicated that the suggested algorithm performed poorly under different lighting conditions. Moreover, The authors Moezi *et al.* [18] used an optimal adaptive interval type-2 fuzzy fractional-order back-stepping sliding mode control method to enhance the performance of the BPS control. The purpose of using the fractional-order back-stepping sliding surface is to reduce the tracking error, while the adaptive interval type-2 fuzzy aims to compensate for the nonlinear effect. The experimental results demonstrated that the proposed method outperformed other types of sliding mode controllers. Finally, Negash and Singh [14] proposed a combination of fuzzy and sliding mode controllers to reduce the chattering effect. The controller was designed in a double loop structure. The outcomes proved the fuzzy logic controller's ability to reduce the chattering impact.

1.9.6 Artificial-Intelligence-Based Controller

Han *et al.* [63] designed a PID Neural Network controller (PIDNN) which was trained using a Differential Evaluation Particle Swarm Optimization (DEPSO) to solve the problem of tracking trajectory. This problem has prompted researchers in [15] to work on where the authors proposed the use of two parallel sub-controllers: base linear controller and NN-based PID with adjustable gains. The latter aims to compensate for the nonlinear effects ignored in the former controller. Both sub-controller outputs are summed together to generate the control signal. The simulation and experimental results demonstrated the effectiveness of the proposed control strategy. On the other hand, A P-PD controller was designed by Gozde [8] where P aims to control the servo motor. Furthermore, the swarm intelligence-based PSO algorithm and the evolutionary algorithm-based Differential Evolution (DE) are chosen to tune the controller's parameters. The findings proved that the PSO algorithm performs better when changes were made to the system parameters. Compared to the DE-based P-PD controller and the classical P-PD controller, the superiority of the PSO algorithm is depicted by smaller overshoot and shorter settling time.

1.9.7 Sliding Mode-Based Controller

The authors in [64] designed an SMC with two methods. The first method is based on the bang-bang compensation mode, where a limitation boundary was assumed for disturbances and uncertainties. In the second method, an SMC with uncertain items to observe compensation was developed. The outcomes showed that the tracking performance was better when using the sliding mode controller with an uncertain item observer, characterized by a small overshoot and a short adjusting time. Another comparative study was conducted by Bang *et al.* [65], who compared the SMC based on error integration and the linear quadratic controller. The outcomes of a circular trajectory showed that SMC outperforms the LQ controller with 4mm and 12mm steady-state error for SMC and LQ controller, respectively. Debono and Bugeja [66] made a comparative study between an SMC and linear state-feedback controller. The results indicated that, compared to the linear state-feedback controller, the sliding mode controller exhibited a faster and more precise response, even at significantly high speeds. In contrast, despite the state-feedback controller being precise, it was found to be slower than the sliding mode controller.

1.9.8 Optimal Controller

Ali *et al.* [67] created a new procedure of an optimal nonlinear controller using the model reference approach. The primary objective of the suggested design is to ensure satisfactory performance even when control loops are coupled and uncertainties exist. The proposed controller's optimal parameters are obtained using the Invasive Weed Optimization (IWO) method, a metaheuristic optimization algorithm. The proposed method demonstrated a good tracking performance despite the presence of disturbances and uncertainties. In [68], Oravec *et al.*simulated the ball's movement with square and circular desired paths when controlled by a Linear Model Predictive Control (LMPC). This simulation was compared to [69], where an Optimal LQ control algorithm was designed. The findings revealed that the MPC algorithm outperformed the LQ control technique in tracking trajectories.

Yuan *et al.* [19] and Awtar *et al.* [70] presented a BPS controller made of a two-loop cascaded strategy, where the inner loop controls the servo motor angle. The outer loop provides an adequate plate angle for the ball's position control. Before any control action, the outer loop needs to adjust the angle of the plate to the desired angle and then measure the ball's position again. This operation requires a fast enough sensor to read the ball's position. In their study, Yuan *et al.* [19] integrated a PID controller in the inner loop and different strategies in the outer loop, such as a fuzzy logic controller supervisor for sliding mode and a

PD controller. Each controller takes approximately 10 seconds to stabilize the ball at a desired position. The experimental results indicated that the resistive touchscreen requires a long time to measure the ball's position, resulting in high overshoot and error during trajectory tracking. The outer loop designed by Awtar *et al.* [70] integrates a lead compensator.

The authors Amin and Ji-Chul [15] and Ali *et al.* [67] adopted different mechanical designs, while Gozde [8] used the most popular mechanical structure that consists of an L-shaped mechanism. This structure transforms a servomotor's rotational motion into a plate's tilting motion. The primary drawback of the L-shaped design is that it overlooks the minor deflection of the plate, which can cause the ball to slip on the plate. Rastin *et al.* [71] used a three Revolute Revolute Spherical (3-RRS) parallel mechanism where the plate is assumed to be its end-effector. This type of mechanical structure is more complex than the L-shaped mechanism since its three axes are dependent and necessitate mathematical computations to derive the appropriate angle for each actuator. Bang and Lee [65] used a 6-DOF Stewart platform to control the ball on its end-effector. Nevertheless, this type of parallel robot used for such applications is over-actuated since the ball-on-plate system requires 4-DOF for optimum control.

Furthermore, two types of sensors were used to locate the ball on the plate: a resistive touch screen [59,70] and a camera that tracks the ball's motion [52,56,61,66,71]. This latter depends on two parameters: The used image processing software that enables the ball to be recognized and the hardware performance of the camera, which consists of the number of Frame Per Second (FPS).
Both parameters could have an impact on the quality of the control. Therefore, in this thesis, we will consider the implemented image processing algorithm and the number of frames per second.

1.10 Conclusion

In conclusion, this chapter has provided a comprehensive overview of parallel robots and their characteristics. By exploring their historical background, fundamental components, and architectures, we have gained valuable insights into the capabilities and advantages of parallel robots, including their high payload capacity, accuracy, and fast task execution. Moreover, we have highlighted representative examples of their application domains, demonstrating the wide-ranging impact of these robotic systems.

Furthermore, this chapter has laid the foundation for our exploration of the ball-on-plate system and its control. By discussing the different approaches to address the control of this system, we have identified the current state-of-the-art strategies and their respective results. This understanding sets the stage for the

subsequent chapter, which will delve into the dynamics of the ball-on-plate system and the kinematics of the 3-DOF parallel robot.

Chapter 2

Modeling of the Ball-on-Plate System and Inverse kinematics

"Life is like riding a bicycle. To keep your balance, you must keep moving."

Albert Einstein

2.1 Introduction

Balancing systems are one of the most challenging problems in the control engineering field and serve as a testing ground for new control design strategies [72]. Systems like the ball and beam system, magnetic levitation, inverted pendulum, and ball on plate system are designed for educational and experimental purposes in laboratory settings. The ball-on-plate system is an advanced version of the ball and beam system, allowing for the manipulation of the ball's position in two directions. Therefore, in this chapter, we begin by providing an overview of the ball-on-plate system structure, including its mechanical and electrical components. Following that, we explore the flow of information within the system and outline the steps involved in deriving its mathematical model. Finally, we identify the servo motor plant and we determine the 3-DOF parallel robot inverse kinematics.

2.2 Structure of the Ball-on-Plate System

The ball-on-plate is a mechatronics system composed of mechanical and electrical parts. It consists of a fixed base and a rigid plate on which the ball rolls freely. Both are connected via at least two legs (links). It typically integrates sensors to detect the ball position and the plate inclination angle, while servo motors or linear actuators are incorporated to move the platform legs. The sensors and actuators work in conjunction to enable precise control of the ball's motion, making them essential components of the system.

2.2.1 Mechanical Structure

The mechanical component of the ball-on-plate consists of two basic parts: a fixed base on which the actuators are installed and attached to the plate by servo rods. The ball-on-plate system structure can be based on many types of parallel robots, like the 2-DOF, 3-DOF, or 6-DOF.

In this chapter, we will analyze two distinct ball-on-plate system versions. First, we will look at the well-known 2-DOF ball-on-plate setup, which is made up of the mechanical components shown in 2.1. This prototype will serve as the foundation for our later examination of the three-dimensional ball-on-plate system. To increase the flexibility of the 2-DOF ball-on-plate system, a third link with a universal joint positioned in the middle of the base and the plate is required. To fully comprehend this system and its components, we will first investigate the various mechanical components of the ball-on-plate system discussed below:

1. The fixed base where the actuators are installed.
2. The plate on which the ball rolls in x and y directions.
3. The third link is called "plate holder" and holds the plate from the center point, providing more flexibility to the platform movements.
4. The servo motor holder is used to fasten the servo motor on the base.
5. The servo motor link converts the servo motor's rotational motion into linear motion.

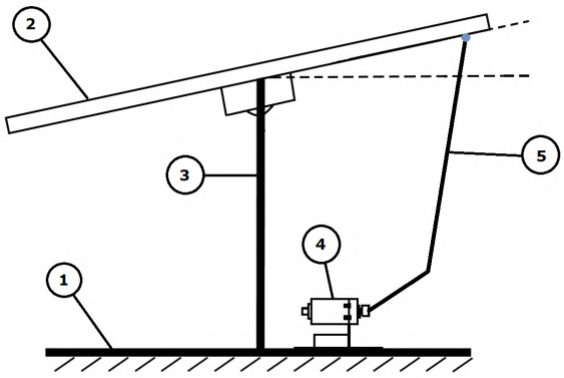


Figure 2.1: Parts of the mechanical structure of the ball on plate system [73]

.

2.2.2 Electrical Structure

The major goal of the ball-on-plate system is to keep the ball in the specified position. Two major electrical components are required to do this: servo motors, which adjust the plate's tilt in response to voltage inputs, and sensors, such as cameras or wire resistive touch screens, which track the ball's movement and transmit its coordinates. Gyroscopes and accelerometers can also be used to determine the orientation of the plate. The sections that follow describe the various electrical components that are utilized for operating the ball-on-plate system.

2.2.2.1 Accelerometer & Gyroscope Sensor

A gyroscope sensor is a tool that measures an object's orientation and can identify the angular velocity as opposed to an accelerometer, which only detects linear motion. Gyroscope sensors are frequently used in

applications where it is challenging for humans to discern an object's orientation. The change in an object's rotational angle per unit of time is expressed in degrees per second. Figure 2.2 shows a sensor that combines an accelerometer and a gyroscope to provide more accurate and robust motion sensing [74].

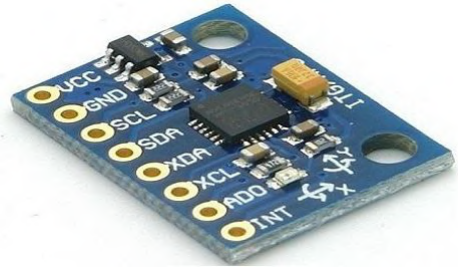


Figure 2.2: GY-521 MPU-605 gyro sensor and accelerometer [74]

.

2.2.2.2 Camera Sensor

The purpose of using a camera is to follow the ball's movement and provide its coordinates in real-time. The procedure starts by capturing and transferring a sequence of images to the processing unit. The latter analyzes the data and detects the ball's position in pixels based on its shape and color.

It is important to note that the higher the camera's resolution, the more detailed the images captured, which can generally cause a longer processing time. It is preferable to allocate a separate processing unit for the processing task. This method frees up more resources for the main processing unit for smooth and real-time control.

2.2.2.3 Wire Resistive Touch Screen

Instead of using a camera to track the ball's movement, a wire-resistive touch screen can be employed as a 2-dimensional sensing device. The touch screen consists of two sheets of material separated by spacers, with a sheet of glass providing a stable bottom layer and a sheet of Polyethylene (PET) as a flexible top layer. When the ball rolls on the touch screen, it presses down the top layer, causing the two resistive surfaces to meet. The position of this meeting can be read by a touch screen controller circuit [75], which measures the voltage variation and converts it into x and y coordinates. These coordinates are then sent via wires to the microcontroller for further processing and analysis, as illustrated in Figure 2.3.

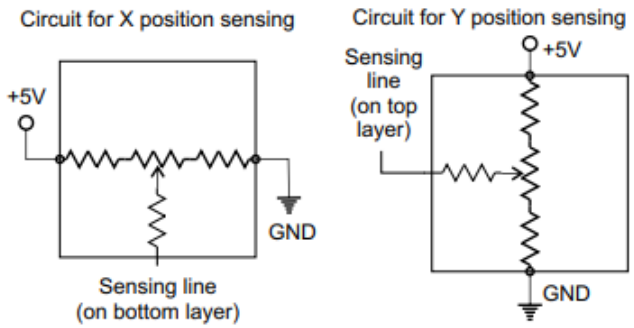


Figure 2.3: Operating principle of a resistive touch screen [75]

.

2.2.2.4 Micro Control Unit

A Micro Control Unit (MCU) is an integrated and compact circuit designed to perform a specific operation. It includes a processor, memory, and both input modules and output modules on a single board or chip. For ball-on-plate systems, there are two types of control. The first consists of using a host computer where the data are received, processed and the decision is determined. In that case, the provided effort is directly sent to the actuators through the microcontroller's output module. In the second type of control, the microcontroller receives the data, processes it and takes a decision, represented by an effort sent to the actuators via the output module. Thus, the microcontroller is autonomous and detached from the host computer.

2.2.2.5 Servo Motor

Servo motors are widely used as actuators for tilting plates through connected links due to their fast and precise response capabilities. These motors consist of a DC or AC motor coupled to a gearbox, which makes them highly efficient and reliable actuators. Additionally, servo motors provide quiet operation and strong anti-load working ability, making them ideal for a variety of applications.

2.3 The Flow of Information

Figure 2.4 illustrates the information flow diagram of a camera-based control system for a ball-on-plate platform. It displays how the x and y coordinates are collected from the camera and sent to the microcontroller. Afterward, based on the implemented algorithm, the microcontroller compares the desired and the

actual ball's positions and provides the effort to the servo motors that tilt the plate to move the ball to the reference position. Ultimately, the ball's actual and desired positions are sent to the PC via the serial port for comparison and analysis purposes.

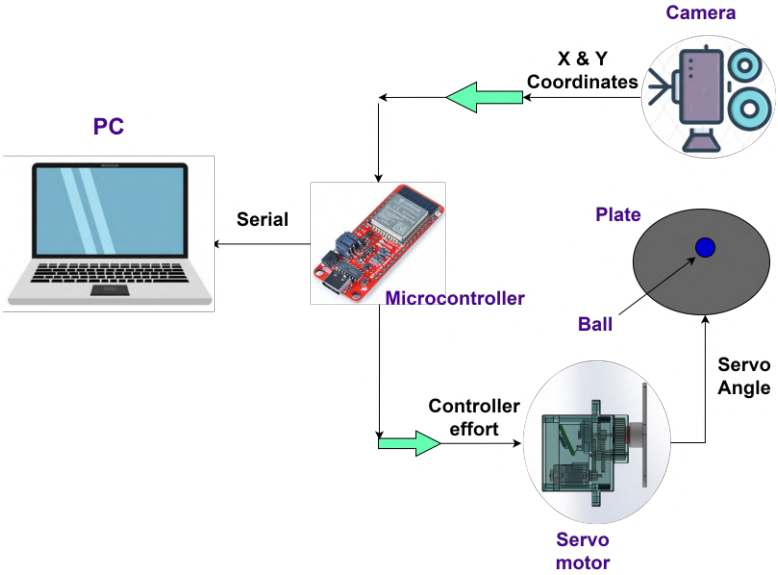


Figure 2.4: Information flow diagram of a closed-loop control system for ball-on-plate platform

2.4 Modeling of the Ball-on-Plate System

Antonito *et al.*utilized the Lyapunov stability theory to verify and demonstrate the stability of the BPS, as presented in reference [76]. The plant parameters of the BPS are described in Table 2.1. Based on the work carried out by Fabregas *et al.* [77], the mathematical model of the BPS is detailed below. The Lagrangian approach is given by:

$$L = T - V \tag{2.1}$$

The kinetic energy T is the sum of the ball's and plate's kinetic energies, which is defined as:

$$T = T_B + T_p \tag{2.2}$$

where,

$$T_b = \frac{1}{2} m_b (\dot{x}^2 + \dot{y}^2) + \frac{1}{2} I_b (w_x^2 + w_y^2) \tag{2.3}$$

Since the ball rolls without slipping, its linear velocities can be expressed as:

$$\dot{x}_b = r_b w_y, \qquad \dot{y}_b = r_b w_x \tag{2.4}$$

Here, r_b and (w_x, w_y) are respectively the ball's radius and angular velocities. Substituting the linear velocities equations in the kinetic energy equation, we obtain:

$$T_b = \frac{1}{2}(m_b + \frac{I_b}{r^2})(\dot{x_b^2} + \dot{y_b^2}) \tag{2.5}$$

The plate's kinetic energy is described by the following expression:

$$T_p = \frac{1}{2}(I_p + I_b)(\dot{\alpha^2} + \dot{\beta^2}) + \frac{1}{2}m_b(x_b\dot{\alpha} + y_b\dot{\beta})^2 \tag{2.6}$$

The potential energy equation of the plate is given by:

$$V = m_b g(x_b \sin\alpha + y_b \sin\beta) \tag{2.7}$$

We substitute Equation 2.1 into the following Euler equation:

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_i} - \frac{d}{dt}\frac{\partial T}{\partial q_i} + \frac{d}{dt}\frac{\partial V}{\partial q_i} = \partial Q_i \tag{2.8}$$

We obtain:

$$(m_b + \frac{I_b}{r_b^2})\ddot{x_b} - m_b(x_b\dot{\alpha^2} + y_b\dot{\alpha}\dot{\beta}) + m_b g \sin\alpha = 0 \tag{2.9}$$

$$(m_b + \frac{I_b}{r_b^2})\ddot{y_b} - m_b(y_b\dot{\beta^2} + x_b\dot{\alpha}\dot{\beta}) + m_b g \sin\beta = 0 \tag{2.10}$$

$$\begin{aligned}\tau_x = (I_p + I_b + m_b x_b^2)\ddot{\alpha} + 2m_b x_b \dot{x_b}\dot{\alpha} + m_b x_b y_b \ddot{\beta} \\ + m_b \dot{x_b} y_b \dot{\beta} + m_b x_b \dot{y_b}\dot{\beta} + m_b g x_b \cos\alpha\end{aligned} \tag{2.11}$$

$$\begin{aligned}\tau_y = (I_p + I_b + m_b y_b^2)\ddot{\beta} + 2m_b y_b \dot{y_b}\dot{\beta} + m_b x_b y_b \ddot{\alpha} \\ + m_b \dot{x_b} y_b \dot{\alpha} + m_b x_b \dot{y_b}\dot{\alpha} + m_b g y_b \cos\beta\end{aligned} \tag{2.12}$$

Designing a controller for the ball-on-plate system is a challenging task due to the presence of nonlinear terms in the mathematical model, which significantly increases the complexity of the controller design

process. Therefore, a simplification is made by disregarding equations 2.11 and 2.12, representing the servo motor torque effect on the system. Equations 2.9 and 2.10 are simplified. Since the plate inclination is small, we assume the following conditions: $\sin\alpha=\alpha$, $\sin\beta=\beta$, and the angular velocities $\alpha=0$ and $\beta=0$. As a result, we arrive at the following set of equations:

$$(\frac{7}{5})\ddot{x} + g\alpha = 0 \tag{2.13}$$

$$(\frac{7}{5})\ddot{y} + g\beta = 0 \tag{2.14}$$

The obtained equations' input and output are respectively the plate inclination and the ball's position, for both X and Y axes. Since the transfer function is assumed to be identical for both axes, only the X-axis will be studied in the following sections. The same approach will be applied to the Y-axis, but its results will not be presented in this thesis.

Parameters	Description
$(\dot{x}_b,\dot{y}_b)$	Ball's linear velocities along x and y axes
r_b	Radius of the ball
I_b	Moment of inertia of the ball
I_p	Moment of inertia of the plate
$(\dot{\alpha}, \dot{\beta})$	Plate's angular velocities
(α, β)	Plate's angle of inclination along x- and y-axes
g	Gravitational acceleration

Table 2.1: Description of the plant parameters

2.4.1 Stability of the Ball-on-Plate System

The system exhibits characteristics of an undamped second-order system. Theoretically, it falls under the category of "marginally" stable; however, practical conditions reveal its inherent instability. Consequently, the design of an appropriate controller becomes imperative to stabilize the system and fulfill the specified performance requirements.

The study in [78] encompassed analyses in both the time and frequency domains. The authors assessed system stability using the Routh stability criterion, revealing a notable absence of sign changes in the first column of the Routh table. Consequently, the findings led to the conclusion that the system is stable after cascading the controller.

2.5 Servo Motor Plant Identification

The proposed methodology aims to approximate the plant model by identifying the transfer function using real-time data. This involves using a computer acquisition arrangement connected directly to the plant. The Data Acquisition System (DAQ) applies an impulse signal to the plant and records the system response as depicted in Figure 2.5. Once the simulation time S_t is complete, the obtained data is then used to build a model of the plant's behavior.

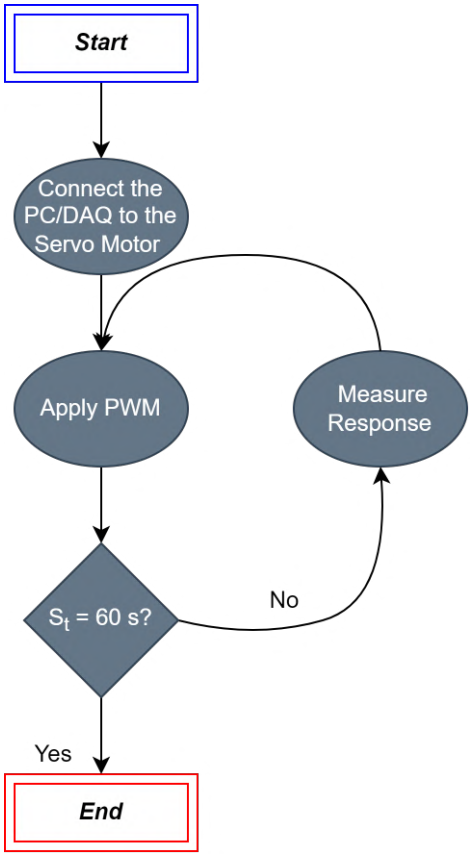


Figure 2.5: Simulink-based identification process

2.5.1 Identification

In this stage, the focus is on estimating the transfer function, which captures the relationship between the output and input signals of the system. The input signal, a 5-volt pulse with modulation, is applied from the computer to the servo motor plant. On the other hand, the output signal represents the motor's speed, measured by the servo motor and transmitted back to the computer [79]. The collected input-output data

patterns are then utilized in the MATLAB system identification toolbox, employing the nonlinear least squares method to estimate the system's transfer function.

After several attempts, among approximately thirty transfer functions, a sixth-order transfer function was identified in this specific experiment. The transfer function, represented by Equation 2.15, exhibited the least amount of error when compared to the actual response of the plant, as depicted in Figure 2.6. The figure showcases the best fittings of transfer functions, with a zoomed-in view highlighting the most accurate transfer function.

$$T_f = \frac{-12.81 - 17.69z^{-1} - 64.81z^{-2} + 7.76z^{-3} - 7.33z^{-4} - 71.96z^{-5}}{1 - 0.3846z^{-1} + 0.2181z^{-2} + 0.1378z^{-3} + 0.58z^{-4} - 0.2559z^{-5} - 0.1082z^{-6}} \tag{2.15}$$

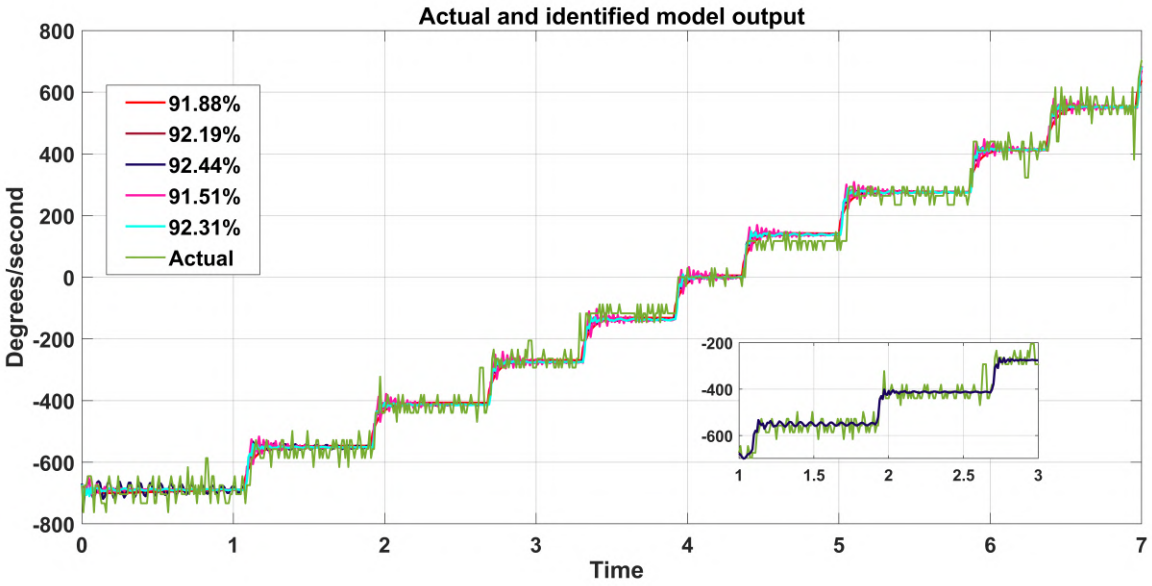


Figure 2.6: Identified transfer function

After identifying the transfer function, we can utilize it in a Simulink environment by disconnecting the servo motor from the PC and selecting the controller structure. Then, we can tune the controller's parameters or test various controllers in a pure simulation environment without risking any damage to the physical plant. This process is referred to as Software-In-the-Loop (SIL).

2.6 Inverse Kinematics of the 3-DOF Parallel Robot

Once we have a thorough understanding of the 2-DOF system, we move on to examining the 3-DOF platform. This more complex version of the system adds another degree of freedom to the setup, providing better control over the ball's motion in terms of accuracy. On the other hand, other mathematical computations should be carried out to control the three actuators, increasing the difficulty of the control system implementation. In parallel robot inverse kinematics, servo motor angles are determined based on the end-effector position. Whereas in our case, we aim to control the ball, not the end-effector. Therefore, we use the ball position to determine the servo motor's orientation. The next mathematical development is based on [1] where Figure 2.7 presents the layout of the servo motors that is useful for calculating the base and plate attachment coordinates as follows:

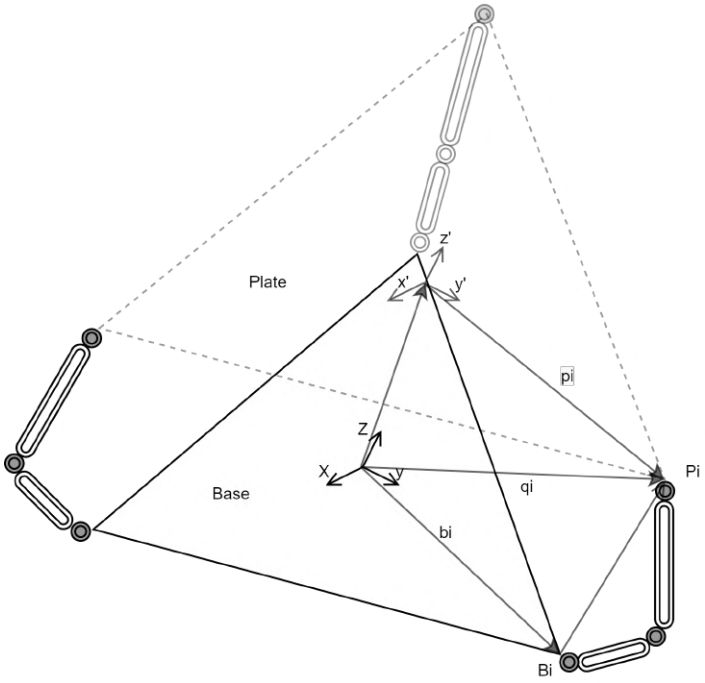


Figure 2.7: A kinematic model of the 3-DOF parallel mechanism [1]

$$b_i = \begin{bmatrix} r_b & \frac{-1}{2}r_b & \frac{-1}{2}r_b \\ 0 & \sqrt{\frac{3}{2}}r_b & -\sqrt{\frac{3}{2}}r_b \\ 0 & 0 & 0 \end{bmatrix} \quad p_i = \begin{bmatrix} r_p & \frac{-1}{2}r_p & \frac{-1}{2}r_p \\ 0 & \sqrt{\frac{3}{2}}r_p & -\sqrt{\frac{3}{2}}r_p \\ 0 & 0 & 0 \end{bmatrix}$$

We compute the rotation matrix for the planned Euler sequence:

$$R = Rx(\theta)Ry(\phi)Rz(\psi) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.16}$$

q_i is the position vector of the connections pi with respect to the base frame:

$$q_i = T + {}^{p}_{b}R \times p_i \tag{2.17}$$

where the translational vector is defined as follows:

$$T = \begin{bmatrix} x \\ y \\ z + h_0 \end{bmatrix}$$

h_0 represents the height of the platform with reference to the base. We compute the necessary legs length for linear actuators as follows:

$$l_i = \sqrt{(q_{0i} - b_{0i})^2 + (q_{1i} - b_{1i})^2 + (q_{2i} - b_{2i})^2} \tag{2.18}$$

Since we use rotational actuators, the servo motor angle equation becomes:

$$\theta = \cos^{-1}\left(\frac{a^2 + s^2 + l_i^2}{l_i^2 + a^2}\right) \tag{2.19}$$

where:

a denotes the length of the servo motor arm.

s indicates the length of the robot leg.

Before we implemented the inverse kinematics algorithm on the microcontroller, we first implemented and simulated it in MATLAB. We carried out this test to avoid kinematics singularities that can cause unwanted behavior or damage to the robot.

2.7 Conclusion

In conclusion, this chapter has provided a thorough overview of the ball-on-plate system, including the mechanical structure and the electrical components. The process, from data collection via the camera to the control of the ball's movement has been explained. Additionally, the derivation and simplification of the ball-on-plate mathematical model have been presented and the servo motor plant identification has been conducted using the experimental data.

A significant milestone in this chapter is the development of the inverse mathematical model for 3-DOF parallel robots that uses three ball joints. This achievement represents a noteworthy contribution as it marks the first time this mathematical model has been successfully determined. This notable contribution unlocks new possibilities and allows us to effectively leverage the characteristics of 3-DOF parallel robots in ball-on-plate systems control. In summary, this chapter serves as a bridge between theory and practice, providing important insights for researchers and engineers engaged in the design and control of ball-on-plate systems.

Chapter 3

Design of a 3-DOF Parallel Robot & Hardware Configuration

"Design is not just what it looks like and feels like. Design is how it works."

Steve Jobs

3.1 Introduction

Robots design and building are crucial tasks before programming them. In this chapter, we will describe the approach used to assure the precision of the mechanical structure's building. Our goal is to use computer-aided design tools to help with the design and construction of the parallel robot outlined in Chapters 1 and 2. We present three key purposes of using computer-aided design tools to attain this goal. First, we use software to design all of the mechanical pieces needed to build the parallel robot. We can generate detailed and precise designs that meet our standards by utilizing the software's sophisticated modeling capabilities.

Second, we use software to simulate and assemble the mechanical pieces, allowing us to evaluate alternate assembly positions and combinations. This virtual assembly method assists in spotting singularities that may occur during the real operation of the parallel robot.

Finally, we use the software to produce the Gcode file, which acts as instructions for the machining operation. During this phase, a CNC machine is used to build the numerous mechanical elements that would eventually constitute the 3-DOF parallel robot. The machining process guarantees that the design is executed precisely.

Following the machining step, we investigate the parallel robot's hardware arrangement. Then, we explore individual components such as servo motors, cameras, and microcontrollers and describe their functions in the entire system. Finally, we analyze the connectivity of various devices, emphasizing the communication protocols used to achieve smooth integration.

3.2 Parallel Robot Design

3.2.1 Computer-Aided Design

Computer-Aided Design (CAD) includes all software and geometric modeling techniques for designing, testing virtually - using a computer and digital simulation techniques - and producing manufactured products [80]. In the following subsection, we describe the chosen CAD software.

3.2.1.1 SolidWorks® Overview

SolidWorks® CAD software is a 3D mechanical design application that allows designers to quickly sketch ideas and experiment with features and dimensions to produce accurate models and drawings [81]. Solid-

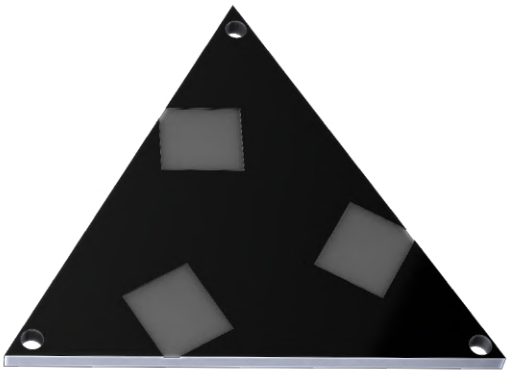
Works® is the most popular software for modeling complex mechanical structures [82]. It allows design engineers to give a real view of what the product will be and to make virtual simulations of the products to reduce the risk of error in production. In the subsequent subsection, we'll use SolidWorks® to design the various mechanical parts that form the parallel robot, experiment with the different features and measurements, and ensure that our final design meets the required specifications.

3.2.2 Design of the Parallel Robot Mechanical Parts

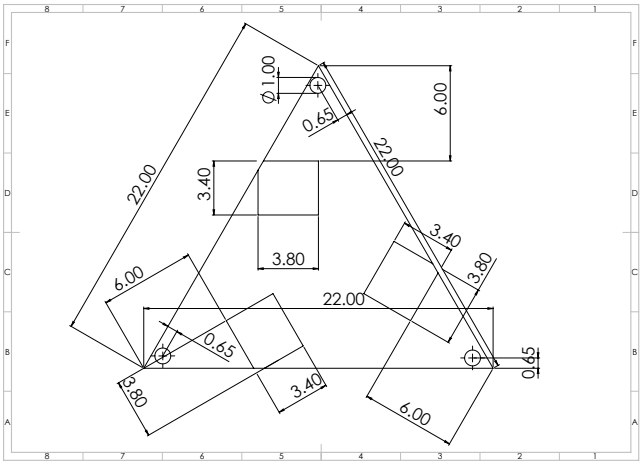
As discussed in chapter 2, the parallel robot consists of electrical and mechanical components, including links, end-effector and servo motor holders. In the following subsections, we will describe the design and function of each of these parts, providing detailed illustrations and specifications where relevant.

3.2.2.1 Base

The base is made of an equilateral triangle with a length of 21 cm. On this base, the locations of the servo motor holders are arranged, with consideration for the distance between each emplacement and the center of the triangle. Additionally, the bolt holes are designed to allow the base to be placed on a larger support for more stability and portability, as depicted in Figure 3.1a. Figure 3.1b accurately illustrates the base dimensions.



(a) Visualization in SolidWorks®



(b) Drawing in SolidWorks®

Figure 3.1: Design of the parallel robot base

3.2.2.2 End-effector

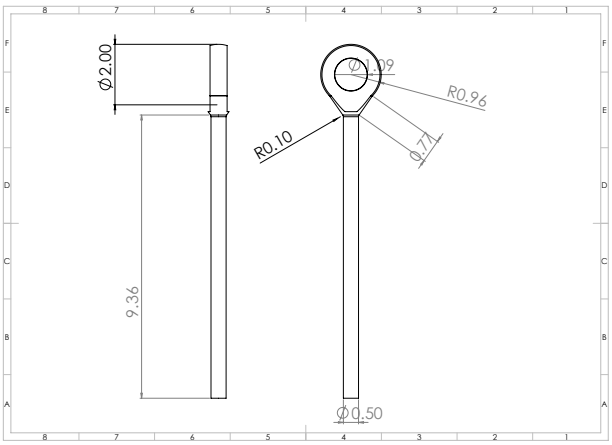
The parallel robot's end-effector is designed as a round plate that allows the ball to roll freely. The plate has a radius of 16 cm, designed with a thickness of 0.4 cm. Furthermore, the bottom of the plate contains three designated locations to join the plate attachments, making it possible to attach the plate to the base, as presented in Figure 3.2.

Figure 3.2: Visualization of the End-effector in SolidWorks®

3.2.2.3 Servo Pushrod

The parallel robot's leg consists of two main parts. The first part is the servo arm, also referred to as the horn, which is connected to the servo shaft and has 6 cm of length. The second part is the pushrod that connects the servo motor horn to the plate via the plate attachment. The servo pushrod is composed of two parts that combine to form an axis of 20 cm long, with two ball joints. Figure 3.3a illustrates one part of the servo motor pushrod, while the drawing is depicted in Figure 3.3b. Additionally, Figure 3.3c shows the servo motor arm.

(a) Visualization of the servo leg



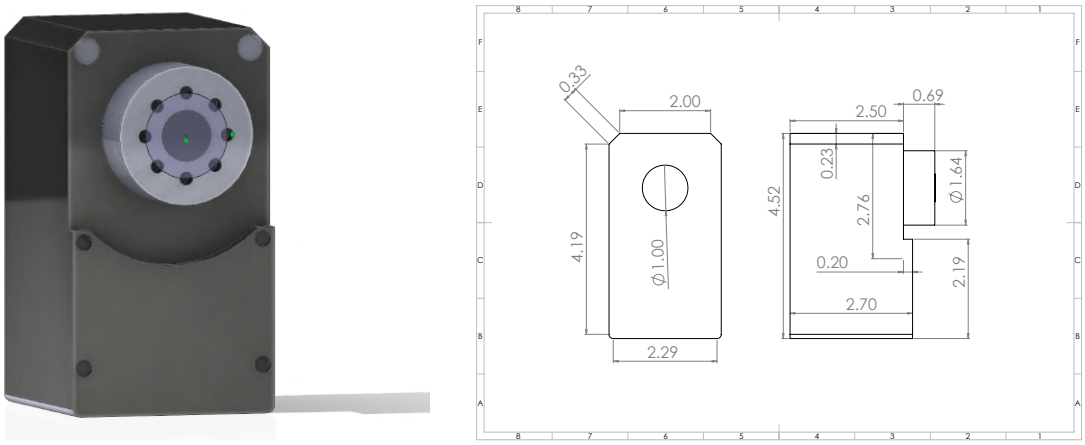
(b) Drawing of the servo leg

(c) Visualization of the servo horn

Figure 3.3: Servo legs design

3.2.2.4 Servo Motor

To accurately design a parallel robot that closely resembles the actual model, it is vital to design the mechanical parts with accurate dimensions, including the servo motor, which must have the same size as the real servo motor for controlling the robot. In light of this, the visual representation of the servo motor design is depicted in Figure 3.4a, accompanied by a drawing illustration presented in Figure 3.4b.



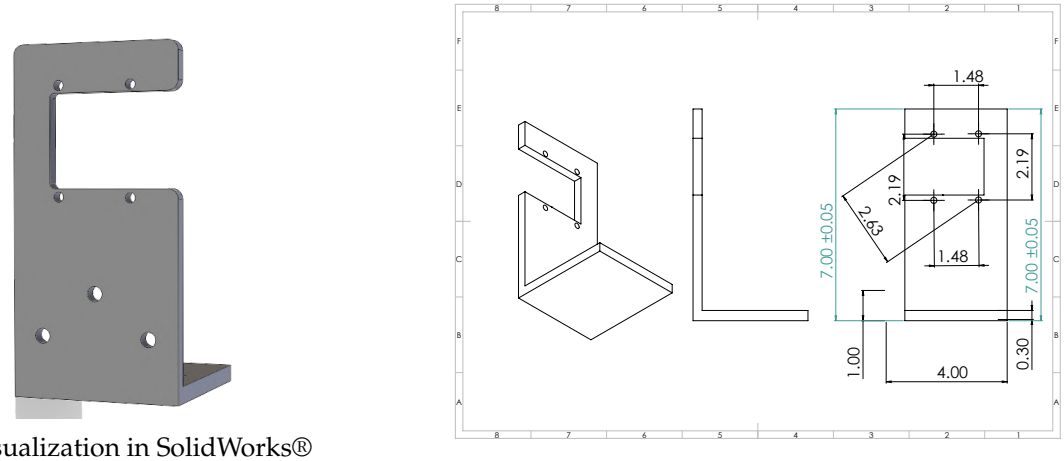
(a) Visualization in SolidWorks®

(b) Drawing in SolidWorks®

Figure 3.4: Design of the servo motor

3.2.2.5 Servo Motor Holder

To securely affix the servo motor to the base, a servo motor holder was designed in a particular manner that enables an oriented position of the actuator, thereby facilitating the control of each link's movement. The servo holder consists of two parts, the first of which secures the servo motor in the desired position using screws. The second component uses bolts to fix the servo in its designated emplacement on the base. Both designed and drawing links are consecutively illustrated in Figure 3.5a and Figure 3.5b.



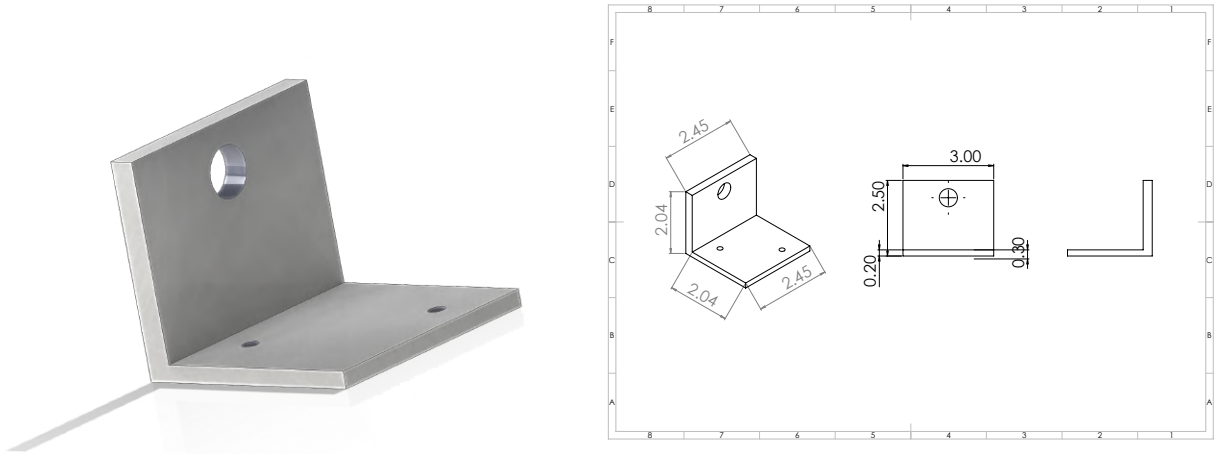
(a) Visualization in SolidWorks®

(b) Drawing in SolidWorks®

Figure 3.5: Servo motor holder design

3.2.2.6 Plate Attachment

The plate attachment plays a crucial role in connecting the servo motor's pushrod to the plate. Its design allows for two distinct faces: the first features two holes that enable the attachment to be securely fastened to its designated position using screws, whose length is less than the thickness of the plate. Meanwhile, the second face includes a hole that facilitates the connection of the ball joint of the servo link using bolts and nuts. Figure 3.6a provides an illustration of the plate attachment. In addition, Figure 3.6b offers a detailed drawing.



(a) Visualization in SolidWorks®

(b) Drawing in SolidWorks®

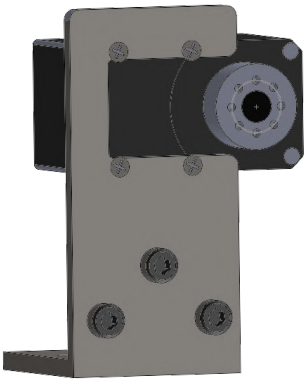
Figure 3.6: Design of the plate attachment

3.2.3 Assembled Parts

When using SolidWorks®, an assembly refers to a digital representation of a mechanism consisting of numerous parts or sub-assemblies. By combining these parts and defining their relationships, the movement of the assembly can be simulated, allowing for visualization in various configurations and situations. This process aids in testing and validating the product before its manufacturing. In our case, once we model the mechanical parts, we assemble them to check the feasibility of placing them in their designed locations with accurate measurements. Therefore, Figure 3.7a showcases the design of the plate assembly, depicting the precise placement of each attachment at a 120-degree angle, while Figure 3.7b illustrates how we attach the servo motor to the servo holder using screws. Additionally, Figure 3.7c gives a clear perspective from the top of the parallel robot, depicting the servo motor configuration's symmetry. Finally, the way that the

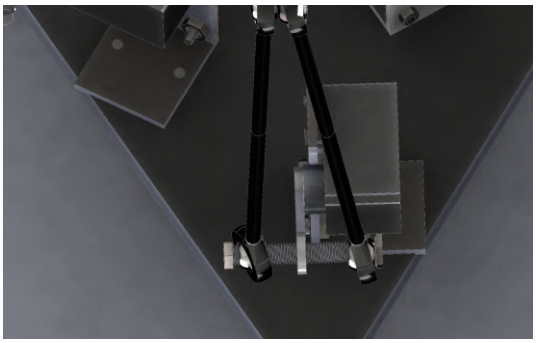
servo links are mounted on the servo motor is portrayed in Figure 3.7d.

(a) Plate and attachments assembly



(b) Front view of the servo motor and holder assembly

(c) Servo motors' layout symmetry



(d) Servo horn part assembled with the servo link

Figure 3.7: Assembly of the different parts of the parallel robot prototype

Figure 3.8 illustrates the design of the fully assembled ball-on-plate system based on a 3-DOF parallel robot. As the various designed parts align correctly with the designated positions, the practical design can be executed without hindrance. To do so, the next subsection discusses the process of machining.

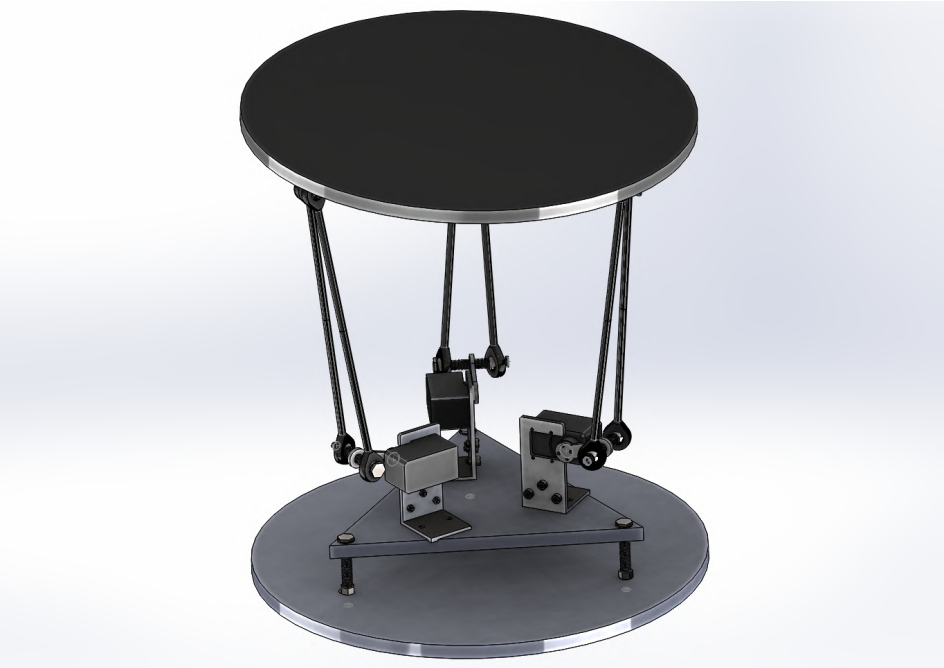


Figure 3.8: Parallel robot: Fully assembled

3.2.4 From Design to Machining: Creating Real Mechanical Parts

Based on our experience, creating Gcode files for a basic CNC machine using SolidWorks® can be challenging. However, for cutting-edge CNC machines, SolidWorks® offers a potent tool with several features that enhance the milling procedure. Since we only had a basic CNC 3018 pro machine, we decided to use a different method for easier handling and superior outcomes. As shown in Figure 3.9, the proposed method consists of two parts. The first part is called preprocessing, which involves generating the DXF file from SolidWorks®. Next, we import the DXF file into the DXF2Gcode software. This program generates the Gcode file based on the defined machining parameters, such as the cutting feed, the machining depth, and the depth per pass. In the second stage, we upload the obtained Gcode file to the Candle software. This software translates the file into machine language that the CNC machine controller can use. Before starting the cut, we define the origins of the 3-axis of the CNC machine.

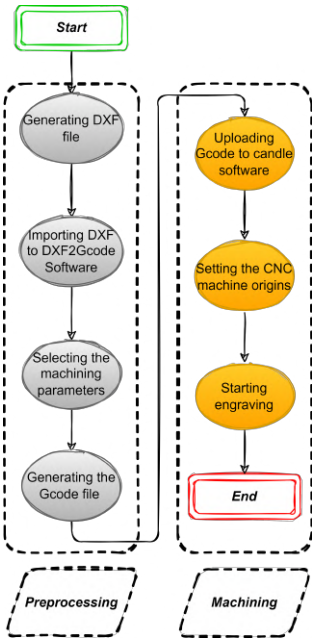


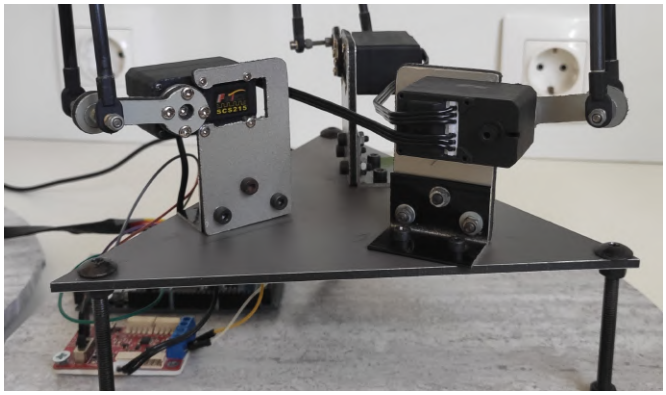
Figure 3.9: Machining process

3.3 Parallel Robot Setup

The top and lower attachments are constructed of aluminum alloy 6061, whereas the servomotor pushrods are made of carbon fiber. The remaining parts are made of aluminum composite, which is renowned for its ability to dampen vibration. Every two rods are attached to a servomotor to strengthen the stability of the plate movement and lessen vibrations. The final design is displayed in Figure 3.10, where Figure 3.10a showcases the overall system. Additionally, Figure 3.10b demonstrates a close-up view of the ball-on-plate system. Furthermore, Figure 3.10c illustrates the practical way that the servo motors are affixed to the triangular base. Lastly, the assembly of the servo motor mounted on the holder is displayed in Figure 3.10d.

(a) Experimental prototype [1]

(b) Parallel robot



(c) Servo motor's layout

(d) Servo motor holder

Figure 3.10: Actual pictures of the proposed ball-on-plate system

3.4 Hardware Configuration

A hardware configuration in a ball-on-plate system refers to the arrangement and connection of physical components such as actuators, sensors, and other mechanical components connected to achieve the desired performance of the system. The hardware configuration can significantly impact the performance and stability of the ball-on-plate system. Therefore careful consideration is often given to this aspect of the design. In our case, we chose a camera, servo motors, and an ESP32 microcontroller.

One advantage of opting for a camera in ball-on-plate systems is that touch panels usually have a square or rectangular shape, which imposes a constraint on the plate's design. In contrast, by using a camera, there is no such limitation, and an end-effector of any shape can be employed. By autonomously processing images via Pixy camera, we reduce the resource consumed by the microcontroller that controls the ball-on-plate system. The control system only receives the simplified data of the ball's coordinates, enabling it to allocate resources toward other tasks. This method leads to more efficient use of the microcontroller's memory and processing power. The used hardware is discussed in the following sections.

3.4.1 Pixy2 Camera

The Pixy2 is a small, quick, and inexpensive camera made especially for robotic applications. A high-performance NXP LPC4330 microprocessor is integrated into the camera, enabling embedded object detection algorithms to track objects. It also allows sending the obtained data in real-time to a host controller. The Pixy2 camera supports several computer programming languages, including Python and C/C++. It can also take and process images quickly with a high frame rate achieving 60 FPS. Below are some of the essential characteristics of the Pixy2 camera:

1. Object recognition: The camera uses image processing algorithms to detect objects based on color, shape, and other characteristics.
2. Line tracking: Pixy2 can track lines, making it useful for line-following robots.
3. Barcode reading: The camera can also read barcodes, which is helpful for inventory management and product tracking tasks.
4. Interfacing options: Pixy2 offers multiple interfacing options, including Universal Serial Bus (USB), Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C) and Universal Asynchronous Receiver-Transmitter (UART), making connecting to various host controllers easy.

5. Open source: Pixy2 is an open-source project, with the software and hardware design available on GitHub.

As depicted in Figure 3.15, we connect the Pixy2 to the host computer via a synchronous serial communication interface that we will explain below.

3.4.1.1 Serial Peripheral Interface

The serial peripheral interface is a synchronous, full-duplex serial communication protocol between controllers (master) and peripheral devices, such as sensors, memories and other Integrated Circuits (ICs). It consists of four signals:

1. MOSI (Master Out, Slave In): The data signal sent from the master to the slave.
2. MISO (Master In, Slave Out): The data signal sent from the slave to the master.
3. SCK (Serial Clock): The clock signal generated by the master to synchronize data transfer.
4. SS (Slave Select): The signal used by the master to select the specific slave device to communicate with.

The SPI's operation principle is based on the exchange of data between a master device and one or more slave devices using four wires. The master device generates the Signal Clock (SCK) and controls the communication by selecting the slave device to communicate using the SS signal. Data is transferred between the master and the slave through the MOSI and MISO signals. During each clock cycle, the master sends and receives a bit of data on the MOSI and the MISO lines, respectively. As shown in Figure 3.11, the data transfer is synced with the SCK, permitting the transfer to continue until all data are sent.

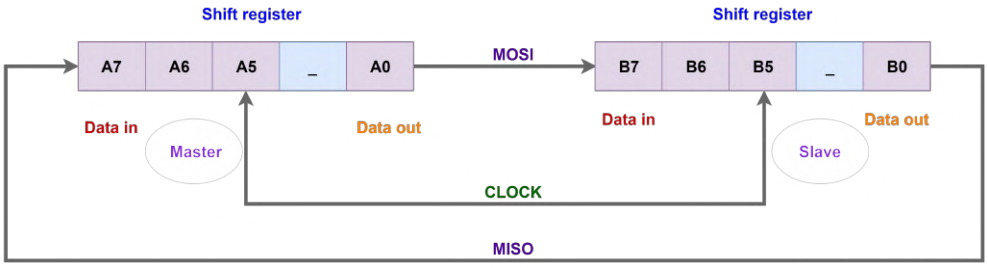


Figure 3.11: Principle of operating

The SPI protocol is designed for use over short distances, typically no more than ten meters [83]. For this reason, we made a cable of 1.3 meters for connecting the camera to the microcontroller, which ensures

that the signal strength is not degraded due to attenuation during transmission. The cable was twisted to reduce the electromagnetic interference and cross-talk between neighboring pairs, as depicted in Figure 3.10a.

3.4.2 Serial Bus Smart Servo Motor SCS15

We chose the serial bus smart servo motor Feetech SCS215 to balance the plate. It is built with metal gear and integrates an STM8 microcontroller with various sensors for providing the following information:

1. Position: The internal servo microcontroller verifies and adjusts the servo motor position, while an analog potentiometer provides the position feedback.
2. Speed: The Feetech SCS215 conveys the speed at which it is rotating.
3. Voltage: The servo reports the voltage range at which it is currently operating. This is crucial for ensuring the servo functions within the manufacturer's recommended limits.
4. Temperature: The servo motor provides its internal temperature to identify potential overheating issues.

The Feetech SCS215 detects potential overloading problems by measuring the load that it is currently carrying. If it exceeds its limited predefined range, the servo motor turns off to prevent it from being overworked, which could cause damage or failure. The SCS215 utilizes an asynchronous packet-based protocol that uses a TTL multidrop bus for connecting up to 254 Servo motor IDs and supporting speeds up to 1Mbps. The servo motor provides a maximum torque of 17 kg.cm with a precision of 0.19 degrees [84]. The SCS215 offers two ways of control: with a Pulse-Width Modulation (PWM) signal to adjust the servo's position, or with a PD controller to move the servo shaft to the desired position with a constant or variable speed, acceleration, and deceleration.

Overall, the Feetech SCS215 is a reliable servo motor well-suited for many robotic and control systems applications [84]. As mentioned earlier, this servo motor uses a packet-based communication protocol, which will be explained in the following subsection.

3.4.2.1 Principle of Operation of Feetech SCS215

This section is based on [85], [84] and [86]. The type of physical connection used by the serial smart servo motor is known as TTL Level Multidrop Bus. This technology connects multiple slave devices in a daisy

chain configuration, where a single master microcontroller can communicate with multiple slave devices. Each servo motor has a unique ID number, allowing the master device to connect with each servo individually.

The communication protocol adopted by smart servo motors is defined as a packet-based protocol, where data is transmitted in discrete packets of fixed size and format. This type of communication protocol includes a packet header, command ID, data payload and checksum. There are two types of packets used in the SCS215 servo motor. The first is called an instruction packet, while the second is known as a status packet, as illustrated in Figure 3.12.

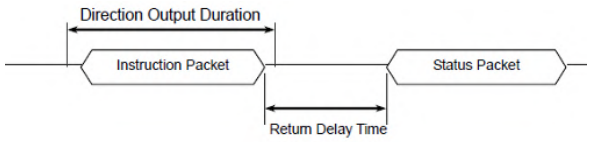


Figure 3.12: Principle of data transmission in Feetech SCS215

The ID is a unique numerical identifier assigned to each Servo motor to distinguish them when multiple servos are connected to a single bus. The main controller can selectively communicate with desired servos by incorporating identifiers in instruction and status packets. Feetech uses asynchronous serial communication with 8 bits, one stop bit and no parity. If multiple Feetechs with the same ID are connected, it will cause packet collision and network issues. Therefore, it is important to assign a unique ID to each Feetech to avoid this problem.

Half duplex UART is a way for devices to communicate with each other using either the Rx or Tx wire, but not both at the same time. It's like a two-way radio where only one person can talk at a time, and the other person must listen. The main controller sets the communication direction to input mode, meaning it's listening for instructions. It changes the direction to output mode to transmit instructions to the Feetech actuators. In half-duplex UART, switching the communication's direction from output to input mode at the right time is necessary. To determine when it's appropriate to make this switch, there are two bits in the UART_STATUS register: TXD_BUFFER_READY_BIT: This bit indicates that the serial transmission buffer is empty and ready to receive new data to transmit. However, it doesn't necessarily mean that all the data previously transmitted has left the Central Processing Unit (CPU). An example highlighting the coding is shown below :

```
1 TxDByte(byte bData)
```

```
  {
    while(!TXD_BUFFER_READY_BIT); //wait until data can be loaded.
    SerialTxDBuffer = bData; //data load to TxD buffer
  }
```

TXD_SHIFT_REGISTER_EMPTY_BIT: This bit is set when all the data in the transmission buffer are transmitted and left the CPU. This bit indicates that switching the communication's direction from output to input mode is safe. This ensures that the CPU is ready to receive new instructions and that the Feetech actuators are ready to respond to them.

```
DIRECTION_PORT = TX_DIRECTION;
TxDByte(0xff);
TxDByte(0xff);
TxDByte(bID);
TxDByte(bLength);
TxDByte(bInstruction);
TxDByte(Parameter0); TxDByte(Parameter1); ...
DisableInterrupt(); // Interrupt should be disabled
TxDByte(Checksum); // Last TxD
while(!TXD_SHIFT_REGISTER_EMPTY_BIT); // Wait till last data bit has been
   sent
DIRECTION_PORT = RX_DIRECTION; // Direction change to RXD
EnableInterrupt(); // Enable interrupt again
```

There is a specific time delay between bytes when sending an instruction packet. If this delay exceeds 100ms, the Feetech actuator will identify it as a communication issue and wait for the packet's following header (0xff 0xff), as explained in Figure 3.1.

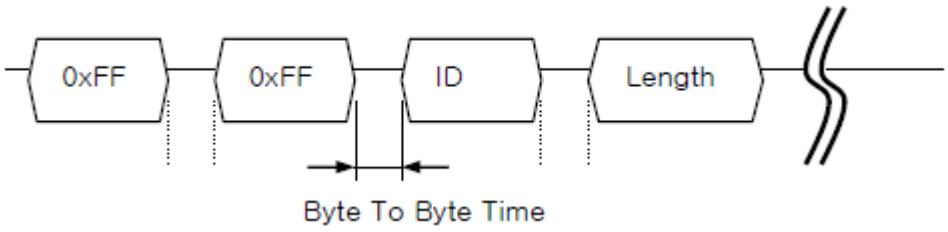


Figure 3.13: Delay time between bytes [85]

An instruction packet is a data packet sent by the host controller to the devices, as depicted in Table 3.1. The headers indicate the start of the packet, whereas the packetID specifies the device ID intended to receive and execute the instruction packet. This field ranges from 0 to 253 (0x00 to 0xFD), providing 254 available ID numbers. Additionally, a Broadcast ID of 254 (0xFE) directs all connected devices to execute the instruction packet. The length field indicates the byte size of the instruction where:

length = number of parameters + 2

Header1	Header2	PacketID	Length	Instruction	Param1...ParamN	ChekSum
0XFF	0xFF	PacketID	length	Instruction	Param1...ParamN	ChekSum

Table 3.1: Instruction packet

The next field specifies the instruction type, with certain types listed in Table 3.2. The auxiliary data field of the instruction is referred to as a parameter, and its role may vary depending on the type of instruction being used. The last field determines if the packet was corrupted during transmission. The Checksum value is obtained using the following formula:

Checksum = (ID + Length + Instruction + Parameter1 + ... + ParameterN)

Value	Instruction	Description
0x01	Ping	To check if the packet was delivered to a device with the same ID as the packet ID
0x02	Read	To read data from the device
0x03	Write	To write data on the device
0x04	Reg Write	To register the packet and put it on standby until it is executed using the action instruction

Table 3.2: Description of the types of instructions

A status packet is a data packet sent by an actuator to a host controller in response to an instruction packet and is depicted in Table 3.3.

Header1	Header2	PacketID	length	Error	Param1...ParamN	ChekSum
0XFF	0xFF	PacketID	length	Error	Param1...ParamN	ChekSum

Table 3.3: Status Packet

In the status packet, the error field indicates any error type that may occur during the servo motor's operation. The errors in Table 3.4 provide information about these potential errors.

Bit	Error	Description
Bit 5	Overload	Current load exceeds torque set
Bit 3	Checksum	Checksum of transmitted instruction packet is incorrect
Bit 2	Overheating	Internal temperature falls outside operating range

Table 3.4: Error status

For example, when the status packet is returned as below:

```
0xFF 0xFF 0x01 0x02 0x24 0xD8
```

The received status packet shows that the error of 0x24 occurs from the actuator whose ID is 01. The binary representation of 0x24 is 00100100, which reveals that both bit2 and bit5 have a value of 1. To clarify, overload and overheating errors have occurred. To verify whether the status packet has been damaged during communication, the checksum is calculated as follows:

Checksum = (ID + Length + Error + Parameter1 + ... + ParameterN)

Below is an example illustrating the operation of the SCS215. The motor receives an instruction from the main controller to send back its current temperature. Our example aims to read data stored in the control table of the SCS215 servo motor. Table 3.5 demonstrates how to send this instruction properly.

Length	Instruction	Parameter 1	Parameter 2
0x04	0x02	Starting Address	Length of Data

Table 3.5: Read instruction

Table 3.6 illustrates how to read the current servo motor temperature at address 43 (0x2B) of the control table of the servo motor having the ID of 1.

Head1	Head1	PacketID	length	Instruction	Param1	Param2	ChekSum
0XFF	0xFF	0x01	0x04	0x02	0x2B	0x01	0xCC

Table 3.6: Instruction packet for reading the servo temperature

Table 3.7 shows the feedback received from the servo motor with an ID number of 1, which indicates that no errors occurred, and the motor reported a value of 32 (0x20) degrees.

Header1	Header1	PacketID	length	Error	Parameter1	ChekSum
0xFF	0xFF	0x01	0x03	0x00	0x20	0xDB

Table 3.7: Status packet for reading the servo temperature

The flowchart representing the communication protocol process of the SCS215 is illustrated in Figure 3.14. The communication starts by sending an instruction packet to the servo controller, which verifies whether the calculated checksum value is the same as the value included in the packet. The servo controller also checks the time interval between packets to ensure it is within a specified range. If the two checksums are not identical or the time interval is too long, the servo controller will not execute the instruction contained in the packet. In some cases, it will notify the main controller about communication issues. On the other hand, if both conditions are valid and depending on the type of instruction, the servo controller either moves the motor to the desired position or sends status feedback to the primary controller, which verifies the checksum value.

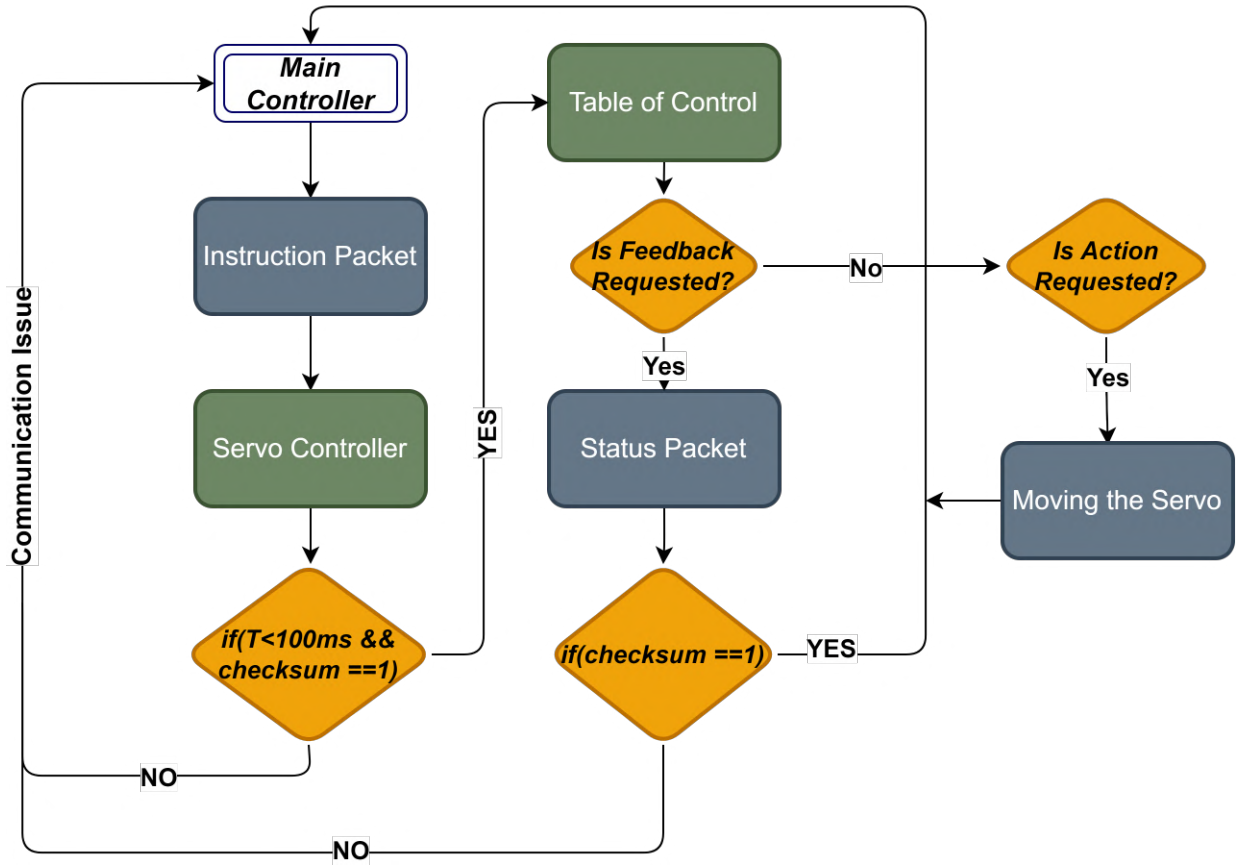


Figure 3.14: A comprehensive overview of the communication process in the SCS215 servo motor

3.4.3 ESP32 Micro Control Unit

ESP32 is a low-cost System-On-a-Chip (SoC) microcontroller with integrated Wi-Fi and dual-mode Bluetooth. It was developed by Espressif Systems, a Shanghai-based Chinese company [87]. According to [88], the microcontroller's features are as follows:

1. Dual Core Processor: ESP32 features two 32-bit cores with adjustable CPU clock frequency from 80 MHz to 240 MHz.
2. Wi-Fi: ESP32 supports 802.11 b/g/n/e/i with various encryption methods.
3. Flash Memory: ESP32 comes with three different sizes of onboard flash memory: 4 MB, 8 MB, or 16 MB.
4. Static Random Access Memory (SRAM): The ESP32 has 520 KB of Static RAM consisting of 320 KB of Dynamic RAM and 200 KB of Intelligent RAM.
5. Input/Output: ESP32 has a wide range of I/O options, including 18 Analog-to-Digital Converter (ADC) channels, 4 SPI interfaces, and 2 I2C buses.

The ESP32 integrates two lower-power Xtensa 32-bit LX6 microprocessors built with a cutting-edge 40 nm process by the Taiwan Semiconductor Manufacturing Company (TSMC). This technology delivers high performance and low power consumption, allowing the ESP32 to be well-suited for a wide range of applications.

3.5 Interconnection Between the Ball-on-Plate System Devices

Figure 3.15 illustrates how the ball-on-plate system devices are interconnected and communicate with one another. Specifically, the Esp32 microcontroller is connected to the host computer via the UART interface, which allows for the transmission of various types of data, such as X and Y coordinates and servo motor information, including temperature, voltage, and position.

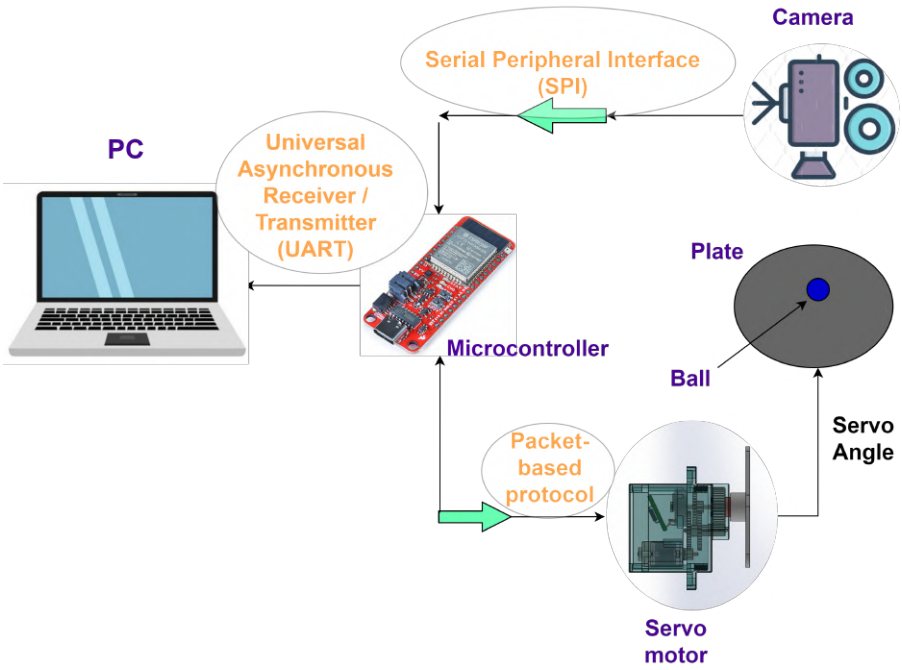


Figure 3.15: Data transmission between the different devices

Moreover, the integrated circuit depicted in Figure 3.16 facilitates the connection between the main microcontroller and multiple motors in a daisy chain configuration. As previously stated, the camera transmits X and Y coordinates to the main controller via the SPI protocol. On the other hand, the servo motors employ a TTL Multidrop communication Bus and packet-based protocol to communicate with the primary controller.

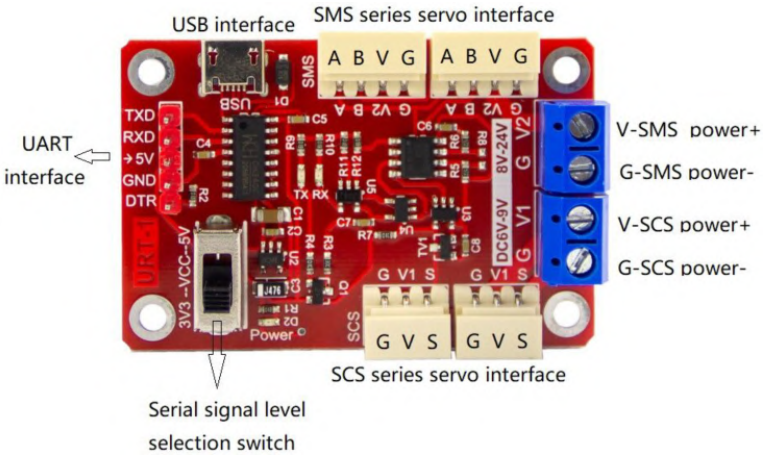


Figure 3.16: Driver for the control of the servo motor [89]

3.5.1 Network On Chip

The NoC architecture interconnects multiple processing elements, memories, and other functional blocks within a System-on-Chip design. The NoC architecture provides a flexible, scalable, and efficient solution for communication within a single chip, enabling the integration of multiple functions into a single compact package [90]. The use of NoC in electronics has become increasingly popular over the last years due to the growing demand for high-performance, low-power, and cost-effective integrated systems. The NoC architecture provides a way to achieve these goals by enabling communication between components at high speed, reducing power consumption, and improving system reliability and scalability [91].

3.5.2 System On Chip

System-on-chip refers to an integrated circuit that incorporates all the components of a system on a single chip. SoCs typically include a CPU, memory (such as RAM and flash memory), peripheral interfaces (such as USB, Ethernet, and UART), and other components, such as Graphics Processing Units (GPUs) and Power Management Units (PMUs).

The purpose of SoC design is to integrate as many components as possible on a single chip to reduce the size, cost, and power consumption. SoCs are used in various electronic devices, such as smartphones and Android Tv boxes, as well as in industrial, automotive, and embedded systems. Integrating multiple components on a single chip provides many advantages, including improved performance and reduced power consumption compared to traditional multi-chip solutions.

3.5.3 Embedded System

Based on the information we have provided in the previous sections, our system is not a NoC system. Our system comprises multiple devices (ESP32 microcontroller, Pixy2 camera, and servo motors) connected through different communication protocols. Although these devices may be connected to form a system, they do not meet the criteria for an NoC system. On the other hand, our system is composed of different devices, each with its own separate integrated circuit. While the ESP32 may be considered an SoC, as it includes multiple components integrated into a single chip, the overall system we described is not a SoC.

A more general term to describe our system could be "embedded system." An embedded system is a computer system integrated into a hardware device to perform a specific task. Embedded systems typi-

cally combine computer software and hardware, such as a microcontroller or microprocessor, memory, and various peripheral components designed to perform a specific task. The term "embedded system" is broad and encompasses a wide range of systems, from simple single-board microcontroller systems to complex multi-board systems with multiple processing units. Accordingly, in our case, we have an embedded system consisting of an ESP32 microcontroller connected to a Pixy2 camera and servo motors through different communication protocols.

3.6 Conclusion

To summarize, this chapter has pointed out the significance of accurate design and manufacture in parallel robots in order to achieve high kinematic precision. To do this, the chapter has been divided into two sections that combine software and hardware procedures.

In terms of software, we have used computer-aided design tools to construct 3D models and simulate the interactions of various mechanical pieces. This method was extremely useful in discovering singularities and improving the design prior to manufacturing. The 3D models developed by CAD acted as blueprints for CNC machines, allowing us to construct the final product with great precision while following the design criteria. This complete software-driven method produced good results and assured the final parallel robot's quality.

Furthermore, the second section of the chapter has delved into the parallel robot system's hardware setup. In this part, we have reviewed the hardware components, such as the servo motor, camera, and microcontroller, as well as the communication protocols used to ensure smooth integration between the various devices. Understanding the hardware configuration and its characteristics is critical for obtaining efficient and dependable control performance. Furthermore, we have defined the distinction between Network on Chip (NoC) and System on Chip (SoC), confirming that the constructed system has embedded system features.

In summary, obtaining optimal control performance in parallel robots requires a combination of accurate design and manufacturing procedures, as well as a grasp of hardware configuration. The following chapter will expand on this foundation by delving into the controller's architecture and the real-time implementation of the suggested algorithm.

Chapter 4

Controller Design and Real-time Implementation

"I hear and I forget. I see and I remember. I do and I understand."

Confucius

4.1 Introduction

To maintain a control system's optimal performance, careful consideration must be given to the quality of its design and the management of latency. Latency, or the delay between triggering a control action and its execution, is crucial in preventing erroneous system behavior. Real-time functionality in the control system is vital since it enables timely responses. However, algorithmic complexity is a major cause of latency and can have a considerable influence on the overall quality of the control system.

In this chapter, we address this challenge by designing an adaptive-neuro fuzzy inference system (ANFIS) based on real data collected from our constructed ball-on-plate system. By leveraging the advantages of ANFIS, we aim to mitigate the adverse effects of algorithmic complexity and minimize latency. Furthermore, we compare the performance of the ANFIS controller with a conventional Proportional-Integral-Derivative (PID) controller to demonstrate the impact of algorithmic complexity on system behavior.

In order to further reduce latency, we suggest a strategy for distributing the implemented algorithm across the two cores of the ESP32 microcontroller. By utilizing the parallel processing capability of the microcontroller, this approach seeks to increase the overall algorithm execution speed and decrease latency.

Finally, we conduct perturbation rejection tests in order to evaluate how well the suggested control technique works. These experiments will demonstrate the robustness and effectiveness of our control strategy by offering quantitative and qualitative insights into the system's response to external disturbances.

The objective of this chapter is to enhance the performance of the control system by resolving the difficulties brought on by latency and algorithmic complexity.

4.2 ANFIS Topology and Controller Design

The double-loop feedback scheme shown in Figure 4.1 is designed to control the servo motor angle in the inner loop and the ball's position in the outer loop. However, the ball and plate system in an open loop is a non-damped second-order system, as observed in [78]. Theoretically, this system can be classified as "marginally" stable when integrated with the servo motor transfer function [11].

To evaluate the stability of the BPS in a closed loop, a Routh stability test is conducted by Garg *et al.* [78]. The authors confirmed that the system is stable after cascading the controller.

The general control scheme of the designed ball-on-plate system is shown in Figure 4.2. It consists of two controllers: one for controlling the ball's position along the X-axis and another for controlling the ball's

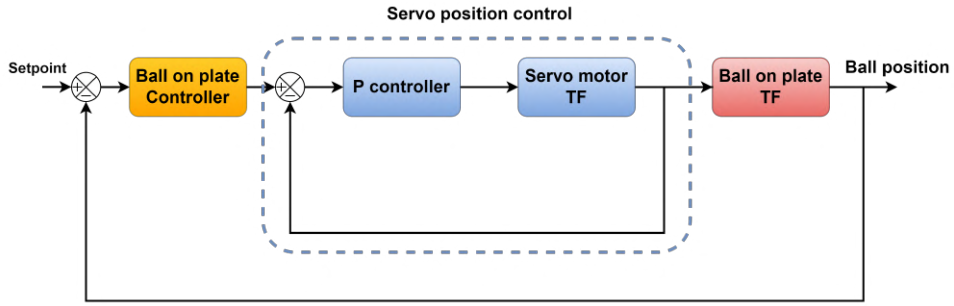


Figure 4.1: Double loop feedback scheme [1]

position along the Y-axis. The outputs of these controllers are then fed to the inverse kinematics algorithm, which determines the angles of the three servo motors.

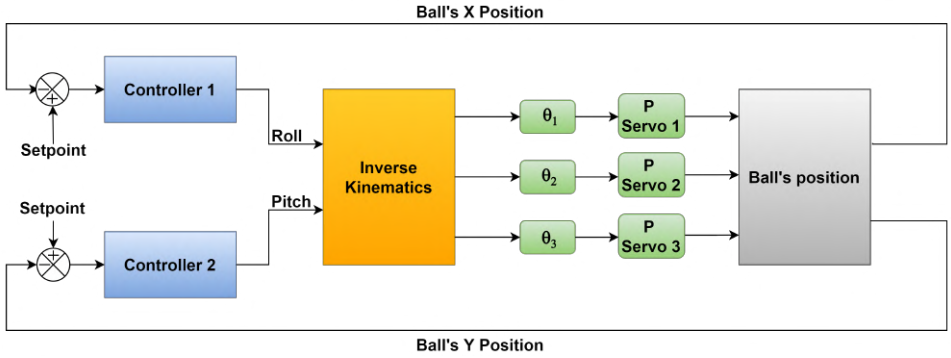


Figure 4.2: The control scheme of the designed parallel robot [1]

4.2.1 Adaptive Neuro-fuzzy Inference System Design

The flowchart of the adaptive network-based fuzzy inference system is depicted in Figure 4.3. The system consists of five layers of neurons, with layer one serving as the input for the model's premise parameters. It takes the input values and passes them forward to the subsequent layers. In layer two, the firing strength of a rule is calculated. This layer essentially determines the influence of each rule in capturing the input-output relationship. Layer three determines the normalized firing strength of each rule. Layer four maps the output of the membership functions, with the parameters in this layer known as the 'consequent parameters.' Finally, layer five generates the control signal. Neurons in this layer combine the outputs from the previous layers to produce the ultimate control signal, representing the model's output. [92].

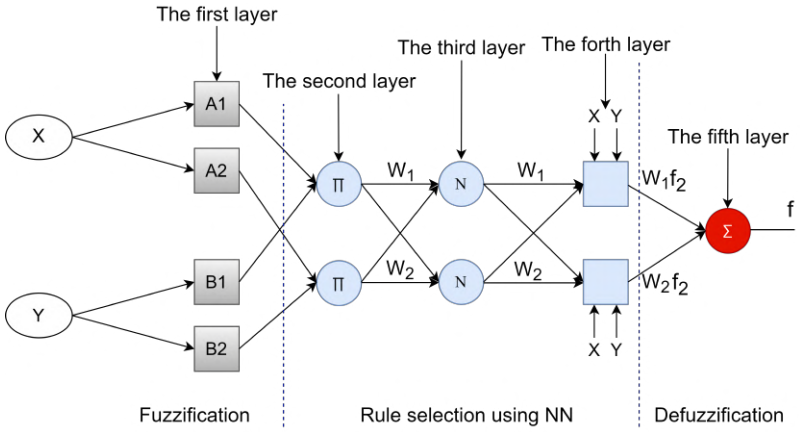


Figure 4.3: Adaptive neuro-fuzzy inference system block diagram [1]

The combination of neural networks and fuzzy logic based on the data set improves and facilitates the design of the controller [93]. The data set consists of two inputs, namely position, and velocity, and one output, representing the plate inclination. For data set collection, a joystick is employed to balance the ball from various locations, and during this process, the position error, position error rate, and plate inclination values are recorded. All the input/output combination values recorded while moving the ball to the desired positions are used to train ANFIS. The training process includes updating the membership function parameters through a hybrid optimization method that combines gradient descent and least squares methods. This approach enhances the controller's decision-making abilities, as it uses real input and output data, resulting in the precise design of the control system.

Figure 4.4: Training performance [1]

Figure 4.4 shows the trained data of the controller's output, where the Root-Mean-Square Error (RMSE) is equal to 0.452 obtained after 305 epochs. Despite the high RMSE value, this data set demonstrated the best performance after undergoing multiple recording operations.

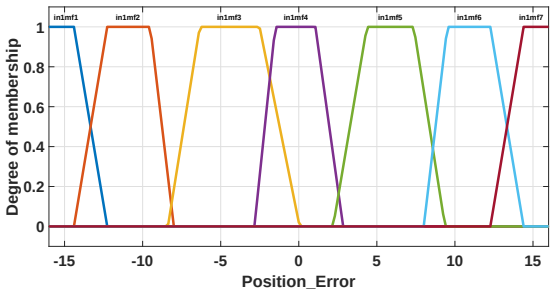


Figure 4.5: Position error membership functions [1]

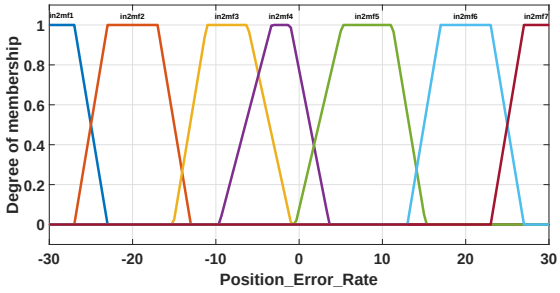


Figure 4.6: Position error rate membership functions [1]

A Sugeno system is often selected for its efficiency in computations. However, with a high number of parameters, overfitting can become a problem. To address this issue, seven membership functions are assigned to each input. Additionally, a trapezoidal membership function type is chosen to enhance the accuracy of the control signal. To illustrate this, Figure 4.5 displays the generated membership functions of the position error. The position error represents the difference between the desired and actual ball positions. On the other hand, Figure 4.6 depicts the membership functions generated for the position error rate. The position error rate measures the ball's velocity.

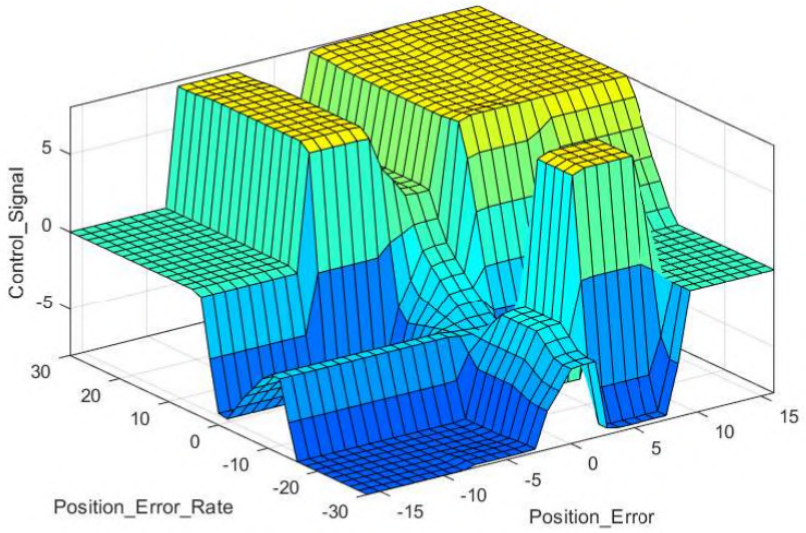


Figure 4.7: Surface of the fuzzy controller [1]

The generated surface of the designed controller can be observed in Figure 4.7. The 3D surface provides a visual representation of the controller's behavior, where its two inputs are limited to a range of -16 to 16,

which represents the radius of the plate, and -30 to 30, which is the ball's velocity. On the other hand, the control signal is limited between -8 and 8. In addition, Figure 4.8 illustrates how the system combines fuzzy logic and neural networks to generate an efficient and adaptable inference engine.

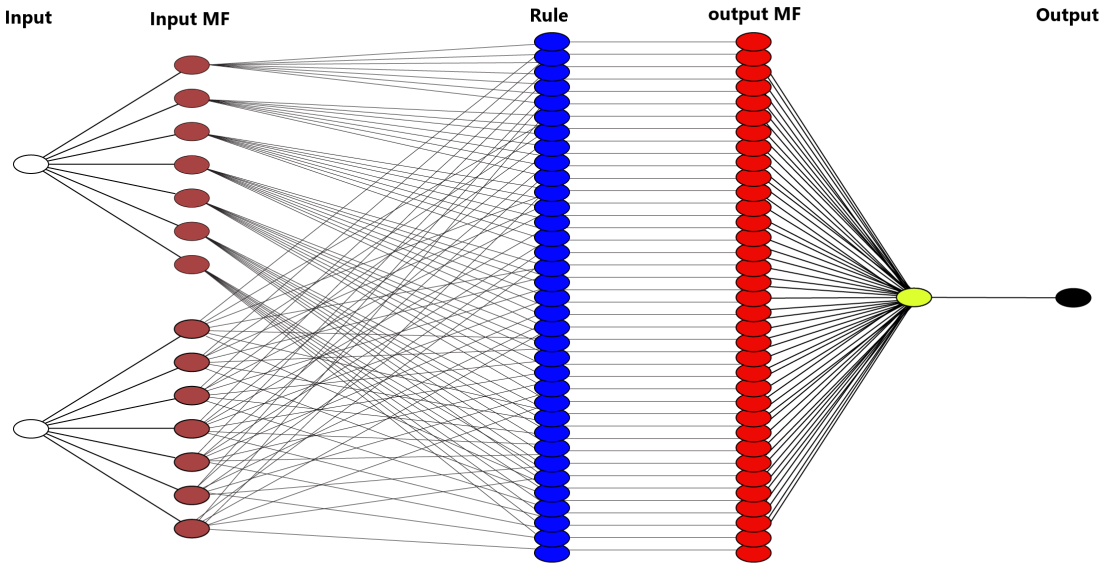


Figure 4.8: The Generated ANFIS structure [1]

4.2.2 PID Controller

The Proportional–Integral–Derivative (PID) controller is widely recognized as the most commonly used controller across various applications. The controller parameters are initially tuned using a PID auto-tuning MATLAB toolbox. Afterward, additional adjustments are made experimentally to optimize the controller's performance. Ultimately, the proportional gain is set to Kp = 3.82, the integral gain is Ki = 2.14, and the derivative gain is Kd = 2.91.

4.3 Experimental Results

Our study aims to implement a real-time fuzzy-based control system. The term 'real-time implementation' refers to the execution of a process or task within a specified amount of time, typically in response to an event or input. Due to various factors, the controllers utilized in our system can potentially consume a

significant amount of the microcontroller's resources, which may negatively impact the processing time and decision-making ability of the control system. Therefore, our focus in the following subsections is to implement a real-time control system that can deliver optimal ball motion performance. We start by describing the algorithmic complexity and the chosen methodology to solve it.

4.3.1 Algorithm Performance Evaluation

Developing code for embedded systems requires more than just making it work. Indeed, it is essential to consider the limited resources available, such as RAM and CPU. However, these limitations do not imply that embedded systems are less efficient than computers and servers. For instance, a high-performance gaming rig may cost $1400, while an embedded system such as PlayStation or Xbox can cost only $500 and offers higher efficiency due to the tight hardware and software integration.

Apple is another concrete example of hardware and software integration. It uses a strategy to optimize software for power, CPU, and RAM consumption. One way to see this by action is by comparing their latest microprocessor Bionic A16, with the Snapdragon 8 Gen 2. Despite the fact that the microprocessor designed by Qualcomm Technologies Inc has a much better hardware performance, the A16 outperformed the Snapdragon in almost all the carried-out tests by users. On the other hand, software like Microsoft Office designed for Windows users could consume more resources on a Lenovo PC than on a Dell PC. The consumption of resources by software is due to its development for a wide range of PCs rather than a specific range.

In addition, an embedded system must be efficient to guarantee consistent results. Consequently, a simple choice of variables can be crucial. For example, using an integer value rather than a character string can have undesirable consequences. Take the instance of the Ariane 5 rocket, which failed a few seconds after Liftoff and cost 500 million dollars to the European Space Agency. This error was due to bad data conversion of the unit of inertia, which gave the horizontal speed of the rocket. This variable was mistakenly converted from a 64-bit format to a 16-bit format, after a few moments, the horizontal speed exceeded 16 bits and caused the explosion of the rocket [94, 95]. Therefore, To evaluate whether the algorithm is resource-consuming or not, we need to :

1- Determine the hardware capabilities. In other terms, how much does the algorithm need to stock data and program code?

2- Calculate the complexity of operations by using Big O notation.

3- Determine the execution time and CPU usage.

In our case, to optimize the programming code, we propose in the subsequent subsections two methods of implementation: the first uses serial programming, whereas the second employs parallel processing.

4.3.2 Serial Processing

"Serial processing" is a form of computing where tasks are executed and completed one at a time. In this approach, the processor carries out each task in sequence. Our proposed method involves Hardware-In-the-Loop testing (HIL) and employs a serial processing approach to execute the algorithm.

To implement this approach, we use a single core (N=01) to carry out each step of the algorithm sequentially. This includes setting up and initializing parameters, collecting pixel data from a camera, converting it to centimeters, and calculating the error position and error position rate. We then pass these values to the fuzzy logic controller to undergo fuzzification, rules evaluation, and defuzzification.

The resulting fuzzy output is converted into precise values, which are fed to the inverse kinematics algorithm to determine the appropriate angle for each servo motor. Remarkably, this process is executed concurrently for both the X and Y coordinates.

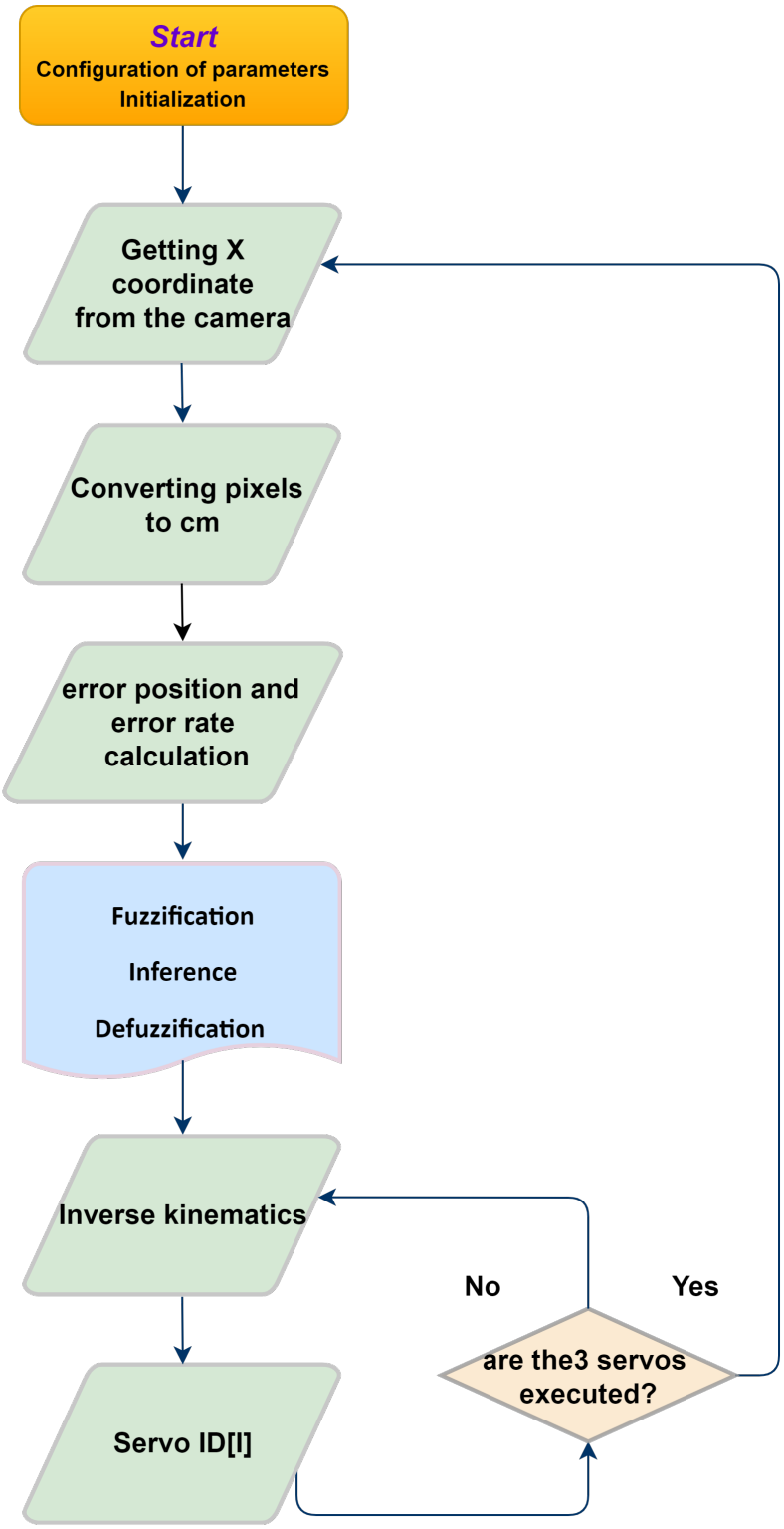


Figure 4.9: Serial processing control loop [1]

4.3.2.1 Decoupling Tasks and Shared Resources

In the second test, we integrate core 0 of the ESP32 microcontroller, making it possible to divide the algorithm into two parts, each in a core. Initially, in core No.0, we retrieve the ball position data from the camera. Then, we convert the obtained values from pixels to centimeters and calculate the position error and position error rate. These two values are fed to the controller which in turn generates the platter orientation angle. This core is called the producer because it produces the roll & yaw angles which are sent to the core N=01, called the consumer. At this level, the calculation of the inverse algorithm is carried out which makes it possible to determine the angle of rotation of each servo motor. This technique allows the sharing of data that is produced from one core and consumed through the second core, as shown in Figure 4.10.

The two cores operate independently, enabling the second core to function autonomously as the first core processes information. This setup allows efficient and rapid data processing, resulting in optimal utilization of controller resources.

The main idea in this part is the choice of tasks allocated to each core. Our decision was based on the algorithmic complexity of the operations, with the two most resource-intensive functions placed in separate cores. The first task, which requires the most resources, involves the execution of two controllers based on inputs, membership functions, rules, and output calculation. The second task is inverse kinematics, which involves exhaustive mathematical calculations that can be taxing on the microcontroller's resources. Therefore, it was allocated to a separate core to ensure efficient processing.

4.3.3 Performance Metrics

Figure 4.11a illustrates the system response of a PID controller, which demonstrates high oscillations and overshoot rate, with a peak of 24.19 cm and a large tolerance band of steady-state error. Moreover, the system's response shows a rise time of 1.32 s, as presented in Table 4.1. Figure 4.11b confirms the long-time adjustment with large servo angle limits.

Figure 4.11c demonstrates the system response of the ANFIS when moving the ball to the origin. The ball oscillates around the set point with a high overshoot rate of 36.9% and a 1.01 s of rise time. In addition, the figure shows that the steady-state error is between a smaller tolerance band when compared to the PID controller. Furthermore, Figure 4.11d and Table 4.2 show that the servo motors rotation angle (degree) is limited to a smaller interval when compared to the conventional PID controller response.

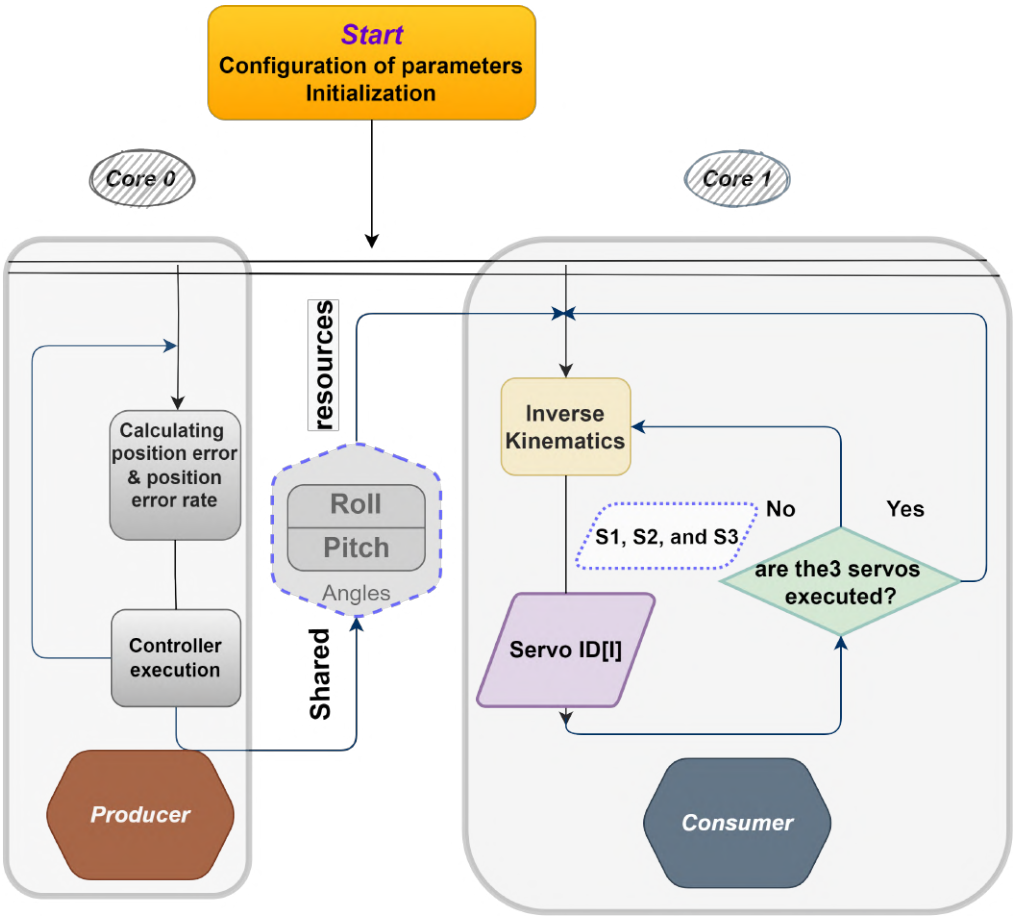


Figure 4.10: Parallel processing unit [1]

Figure 4.11e shows the system response of a PID controller using both microcontroller cores. It demonstrates a high overshoot rate, with a peak of 26.58 cm and a rise time of 1.42 s. Moreover, the figure depicts that the steady-state error ranges between a small tolerance band of 2 cm and 2.5 cm. Additionally, Figure 4.11f exhibits the range of the servo motors angle, which begins at a large limit and decreases gradually.

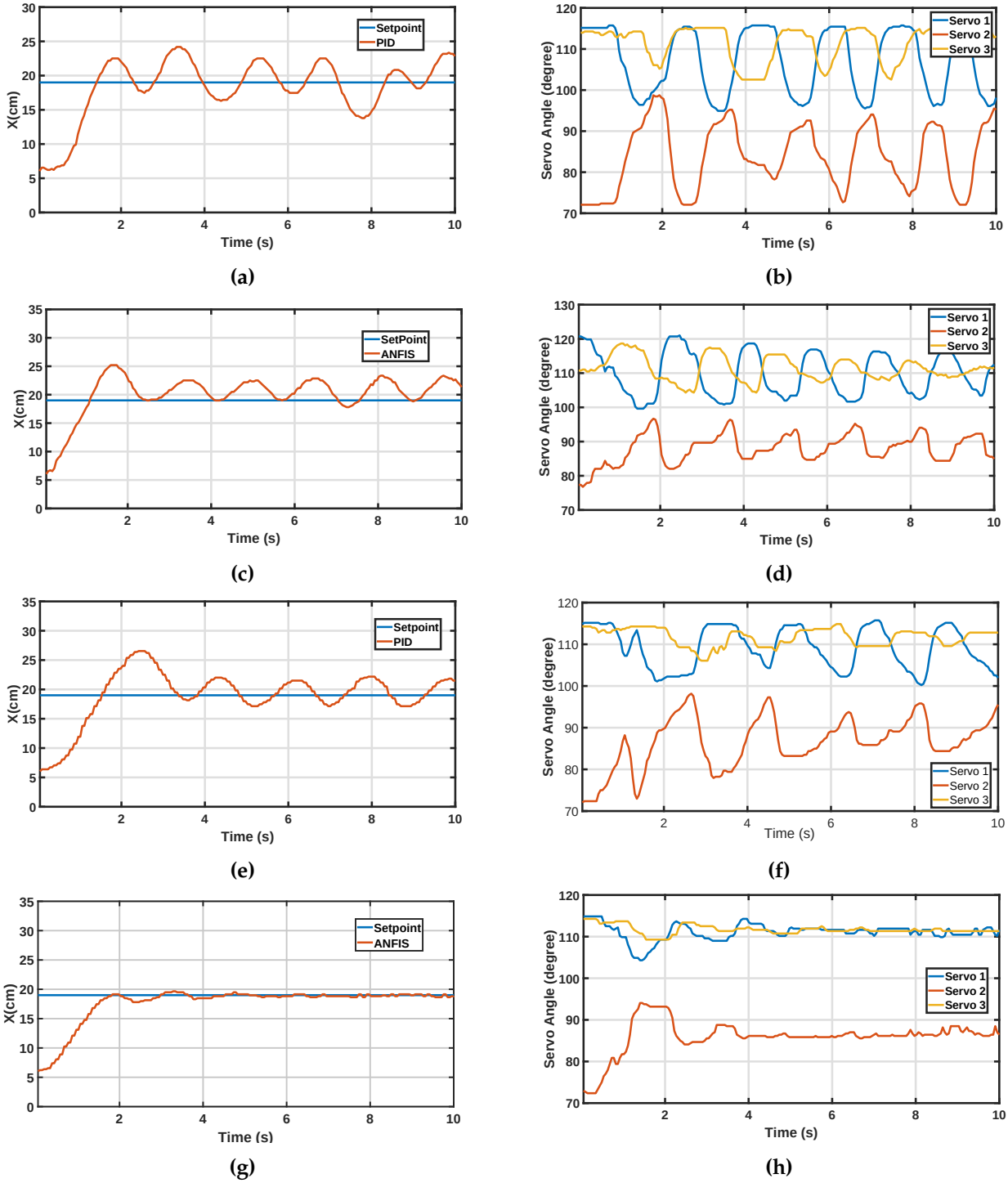


Figure 4.11: Settling the ball to the origin using PID/ANFIS on dual and single-core [1]

Figure 4.11g illustrates the system response of the ANFIS using both cores of the ESP32 microcontroller

when taking the ball to the origin. It shows a low overshoot rate with a rise time of 1.44 s. Moreover, the ball reaches the desired point smoothly, with no oscillations and no steady-state error. On the other hand, Figure 4.11h demonstrates the range of rotation of the servo motor, which is limited to a small interval. This means the ball reaches its position with smooth movement and minimal effort.

Table 4.1: Performance metrics comparison between both controllers in single and dual-core [1]

	Single-core			Dual-core		
Controllers	Rise time (s)	S-S error (cm)	Over-shoot (%)	Rise time (s)	S-S error (cm)	Over-shoot (%)
PID	1.32	-5 to 5	18.4%	1.42	2 to 2.5	40.27%
ANFIS	1.01	0 to 4	36.9%	1.44	0	0.423%

We can conclude that when using a single-core architecture, the control system misses some real-time ball positions and speeds, causing the microcontroller to react abruptly and resulting in small rise time and large oscillations like when using both PID and ANFIS in single-core architecture. We resolved that issue by implementing the algorithm in the dual-core architecture, which allowed the execution of the code with much improved speed, enabling the controller to capture the ball movements more accurately and provide the appropriate control signal in real-time. Therefore, we were able to achieve the actual performance of our designed controller through this configuration.

In addition to settling the ball to the origin test, the rejection disturbances test is also evaluated for both controllers using single- and dual-core architectures. Figure 4.12 shows the performance of each controller for single- and dual-core execution when the ball is moved away from the origin. The PID controller demonstrates a high damping ratio when using a single core or a dual-core in a smaller interval, which keeps oscillating around the set-point. On the other hand, the ANFIS exhibits a good perturbation rejection when using a single core and an excellent result when using a dual-core; it rejects perturbation within 4 seconds with no steady-state error. Thus, The parallel scheme we have chosen has improved the efficiency of the implemented algorithm by having loops that run faster. This technique allowed the controller to run in real-time.

Table 4.3 presents the algorithm processing time, revealing that the ANFIS dual-cores implementation method significantly improved the control system's execution speed. This enhancement led to a reduction in processing time, with a noticeable difference of 30191 μs when compared to the single-core implementa-

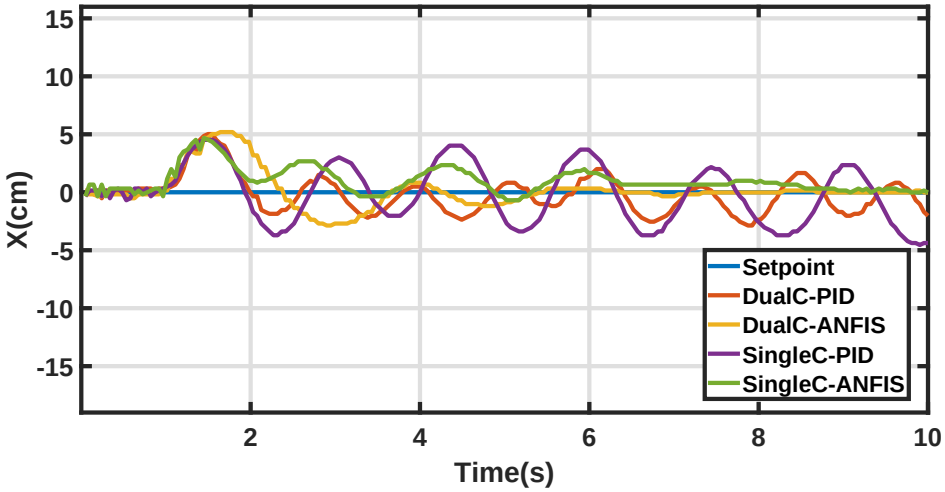


Figure 4.12: Disturbance rejection using PID and ANFIS on single and dual-core [1]

Table 4.2: Servo motor excursion limits [1]

Controller	Single-core	Dual-core	
PID	S1=[95-115], S2=[72-98], S3=[102-115]	S1=[110-115], S3=[220-115]	S2=[85-95],
ANFIS	S1=[100-120], S2=[100-120], S3=[85-95]	S1=[115-105], S3=[115-110]	S2=[72-93],

tion technique. However, it is worth noting that the PID controller did not demonstrate any improvement when utilizing the dual-core implementation method. This lack of improvement can be attributed to the PID controller's inherent simplicity and compact size in its implementation.

Table 4.3: Algorithm Execution Speed

Controller	Single-core	Dual-core	Speedup
PID	31975 μs	30627 μs	4.2%
ANFIS	127199 μs	97008 μs	23.7%

4.4 Conclusion

Finally, this chapter provided a study that compared the performance of an ANFIS controller with a traditional PID controller in operating a 3-DOF parallel robot-based ball-on-plate system. As discussed in Chapter 3, the ball-on-plate system was designed and manufactured using SolidWorks® and a CNC machine.

The study was carried out in several stages, beginning with offline training, in which a dataset was obtained and utilized to train the ANFIS controller. The gradient descent and least squares approaches were used in the training phase. Following that, the ANFIS structure and inverse kinematics were implemented on the ESP32 microcontroller in C++.

The performance of the ANFIS controller was then compared to that of the PID controller. The results revealed that the ANFIS controller outperformed the PID controller in terms of rise time and steady-state error. The utilization of ANFIS led to the stable and smooth movement of the ball, exhibiting minor overshoot and no steady-state error.

To further enhance the system's response, the implemented algorithm was divided between the two cores of the ESP32 microcontroller. This distributed implementation effectively reduced the load on a single core, resulting in reduced latency and improved system performance. The empirical evaluation of these control techniques demonstrated the importance of parallel processing, particularly in scenarios with computational resource constraints.

Moreover, the disturbance rejection test provided further evidence of the benefits of parallel processing in achieving robust system behavior. It is worth noting that while the ANFIS controller showed significant performance improvements, the use of a simpler PID controller did not yield substantial enhancements, despite consuming fewer computational resources.

This chapter contributes to the field of control system design for the ball-on-plate system by comparing the ANFIS and PID controllers and studying the benefits of distributed processing. The findings emphasize the benefits of using sophisticated control approaches such as ANFIS and parallel processing to provide improved control performance, particularly in systems with complicated dynamics and demanding performance constraints.

General Conclusion and Perspectives

"All progress is precarious, and the solution of one problem brings us face to face with another problem."

Martin Luther King, Jr.

As control systems continue to be used in industries such as robotics, agriculture, and manufacturing, and embedded systems become more common in applications such as smart homes and automobiles, the market needs for high-performance systems grows. Meeting these objectives, however, is becoming more difficult as these systems get more complicated, with more advanced functionality and resource requirements. Further progress in the development of efficient real-time implementation approaches can address these challenges and open a wide range of new advanced features integration for a more smart environment. This step will consolidate the substantial progress made in recent years and pave the way for novel and innovative applications of control systems.

In this thesis, we have successfully designed and constructed a parallel robot using cutting-edge design and machining tools. Additionally, we have demonstrated the data-driven control systems' capacity to overcome design challenges such as uncertainty and nonlinearity. This method has helped us to save time and effort when designing controllers for systems involving uncertain and nonlinear factors. Finally, we have successfully proved the performance of the suggested implementation approach using two evaluation tests, demonstrating that it is only efficient for systems that use all of the microcontroller's resources. Furthermore, the mechanical structure we have developed can be used as a test bed for future experiments and assessments, particularly in educational settings where students can gain a better understanding via hands-on experiences and the learning-by-doing method. Furthermore, these findings open up new opportunities for future studies and advancement in this field, such as exploring the following points:

- Evaluate both single-core and parallel programming techniques' power consumption, in order to determine which approach is more economical.
- Further testing new approaches for control such as deep reinforcement learning which can learn from wrong ball movements to enhance the controller's decision.
- Developing and evaluating new microcontroller's performance, such as the STM32 using efficient and robust control algorithms for different balls' sizes and weights.
- Exploring the use of deep learning and artificial intelligence for improving the control system performance and adaptability.

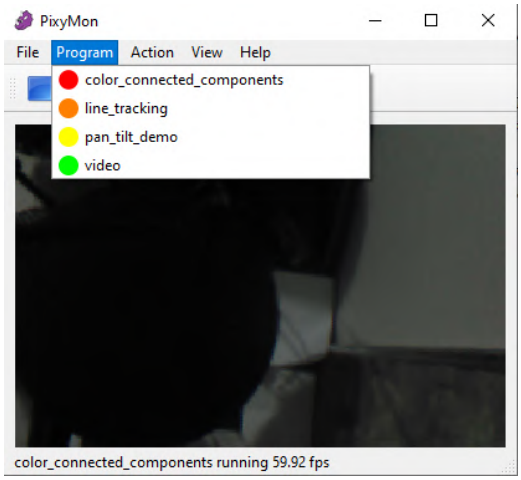
In conclusion, The results of this study provide new avenues for future advancements in control and embedded systems to meet the constantly evolving market demands for high-performance systems.

Chapter 5

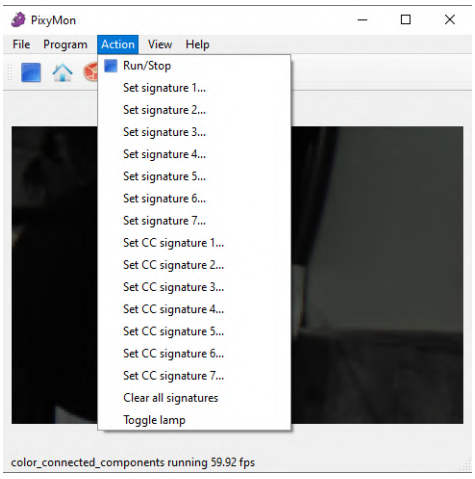
Appendix A

5.1 PixyMon V2 Software

Before using the Pixy camera, we must first start with its training. The training of the camera is done by two methods: using the PixyMon V2 software or by using a button placed on the top of the camera. In this section, we will present the features of the PixyMon V2 software that is depicted in Figure 5.1. PixyMon V2 provides a friendly interface where we can train the camera to recognize barcodes, objects, and lines. It also allows the definition of the data transmission baud rate and the communication protocol used for that goal. Furthermore, PixyMon V2 software makes it possible to label up to seven recognized objects.



(a) Pixy features



(b) Pixy's labels' window

Figure 5.1: Examples of the Pixy2 camera software interface

To utilize the Pixy2 camera effectively, we first extract the object's coordinates in pixels. However, since we need to work with centimeters for calculating the inverse kinematics mathematical model, a conversion from pixels to centimeters is necessary.

To facilitate this conversion process, we began by securing the camera on a stable holder. The Pixy2 camera's video frame size is 315 pixels in width and 206 pixels in height, resulting in a matrix that represents the camera's field of view. In the camera's reference frame, the pixel coordinates of an image follow a convention where the first pixel is located at the upper-left corner. The x-coordinate increases from left to right, while the y-coordinate increases from top to bottom. Consequently, the far lower-right position of the image would have pixel coordinates (x = 315, y = 206).

Afterward, we proceed by measuring the width and height of the frame using a ruler. The width of the frame measures 53.8 cm, while the height measures 30.8 cm. With these measurements in hand, we can now determine the conversion factor for our calculations as follows :

For the X-axis:

$$\frac{52.9cm}{315pixels} = 0.168 \tag{5.1}$$

For the Y-axis:

$$\frac{31.7cm}{206pixels} = 0.154 \tag{5.2}$$

5.2 SCS Feetech Servo Motor Software

In the thesis project, the servo motors played a crucial role. Initially, each servo motor was programmed with a unique ID and a specific transmission baud rate. The debug interface, as shown in Figure 5.2, provided several functionalities:

1. The ability to search for the connected servo motors and identify their ID.
2. Control of servo motors using different control techniques, including adjustable speed and acceleration.
3. Retrieval of the servo motor's feedback, such as position, torque, speed, etc.

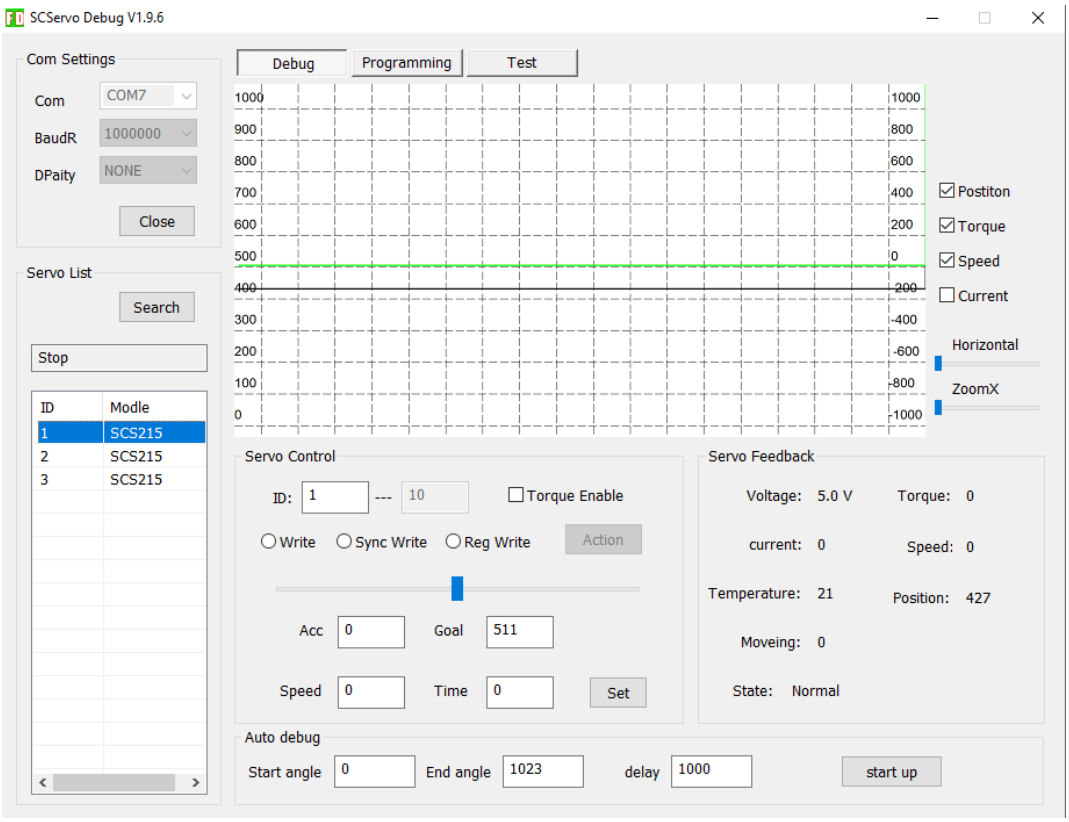


Figure 5.2: The SCS215 debug window

The second interface which is depicted in Figure 5.3 shows the programming window allowing for modifying the various SCS215 parameters. These parameters included integrated PID controller parameters, maximum allowed temperature, servo motor ID, and more. Additionally, this interface facilitated firmware updates whenever new versions were available.

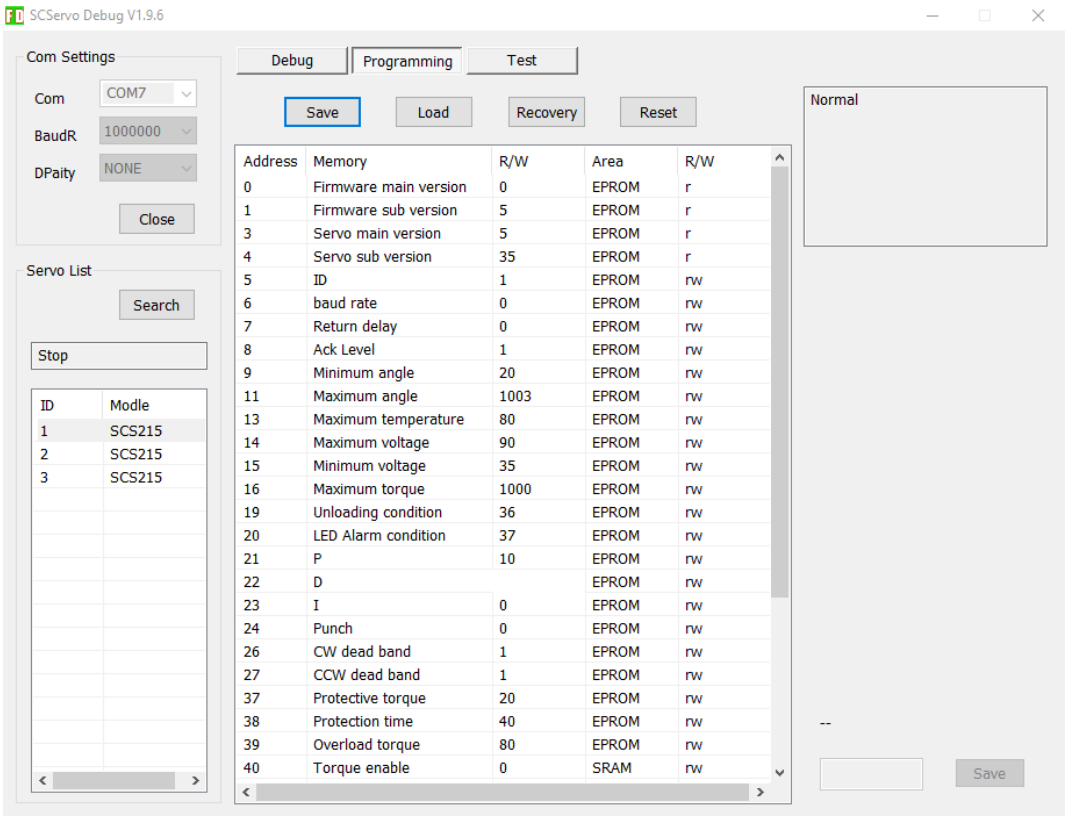


Figure 5.3: The SCS215 programming window

Bibliography

[1] Oussama Hadoune and Mohamed Benouaret. Anfis multi-tasking algorithm implementation scheme for ball-on-plate system stabilization. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 10(4):983–995, 2022.

[2] James R Powell. The quantum limit to moore's law. *Proceedings of the IEEE*, 96(8):1247–1248, 2008.

[3] John Shalf. The future of computing beyond moore's law. *Philosophical Transactions of the Royal Society A*, 378(2166):20190061, 2020.

[4] Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.

[5] Chuen-Chien Lee. Fuzzy logic in control systems: fuzzy logic controller. i. *IEEE Transactions on systems, man, and cybernetics*, 20(2):404–418, 1990.

[6] Guo-Qiang Zeng, Xiao-Qing Xie, Min-Rong Chen, and Jian Weng. Adaptive population extremal optimization-based pid neural network for multivariable nonlinear control systems. *Swarm and evolutionary computation*, 44:320–334, 2019.

[7] Vladimir Tudić, Damir Kralj, Josip Hoster, and Tomislav Tropčić. Design and implementation of a ball-plate control system and python script for educational purposes in stem technologies. *Sensors*, 22(5):1875, 2022.

[8] Haluk Gözde. Comparative analysis of evolutionary computation based gain scheduling control for ball and plate stabilization system. *Balkan Journal of Electrical and Computer Engineering*, 7(1):44–55, 2019.

[9] J.-P. Merlet. *Parallel robots*. Number 74 in Solid mechanics and its applications. Kluwer Academic Publishers, Dordrecht ; Boston, MA, 2nd ed edition, 2006.

[10] Aghiad Kassem, Hassan Haddad, and Chadi Albitar. Commparison between different methods of control of ball and plate system with 6dof stewart platform. *IFAC-PapersOnLine*, 48(11):47–52, 2015.

[11] Emmanuel Okafor, Daniel Udekwe, Yusuf Ibrahim, Muhammed Bashir Mu'azu, and Ekene Gabriel Okafor. Heuristic and deep reinforcement learning-based pid control of trajectory tracking in a ball-and-plate system. *Journal of Information and Telecommunication*, 5(2):179–196, 2021.

[12] Faiber I Robayo Betancourt, Samuel M Brand Alarcon, and Luisa F Aristizabal Velasquez. Fuzzy and pid controllers applied to ball and plate system. In *2019 IEEE 4th Colombian Conference on Automatic Control (CCAC)*, pages 1–6. IEEE, 2019.

[13] Kittipong Yaovaja. Ball balancing on a stewart platform using fuzzy supervisory pid visual servo control. In *2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA)*, pages 170–175. IEEE, 2018.

[14] Andinet Negash and Nagendra P Singh. Position control and tracking of ball and plate system using fuzzy sliding mode controller. In *Afro-European conference for industrial advancement*, pages 123–132. Springer, 2015.

[15] Amin Mohammadi and Ji-Chul Ryu. Neural network-based pid compensation for nonlinear systems: ball-on-plate example. *International Journal of Dynamics and Control*, 8(1):178–188, 2020.

[16] Yongyut Pattanapong and Chirdpong Deelertpaiboon. Ball and plate position control based on fuzzy logic with adaptive integral control action. In *2013 IEEE International Conference on Mechatronics and Automation*, pages 1513–1517. IEEE, 2013.

[17] Jie Ma, Hao Tao, and Jingwen Huang. Observer integrated backstepping control for a ball and plate system. *International Journal of Dynamics and Control*, 9(1):141–148, 2021.

[18] Seyed Alireza Moezi, Ehsan Zakeri, and Mohammad Eghtesad. Optimal adaptive interval type-2 fuzzy fractional-order backstepping sliding mode control method for some classes of nonlinear systems. *ISA transactions*, 93:23–39, 2019.

[19] Dongfeng Yuan and Zhenhao Zhang. Modelling and control scheme of the ball–plate trajectory-tracking pneumatic system with a touch screen and a rotary cylinder. *IET control theory & applications*, 4(4):573–589, 2010.

[20] Grigore Gogu. *Structural synthesis of parallel robots*. Number v. 149, 159, 173, 183 in Solid mechanics and its applications. Springer, Dordrecht ; London ; New York, 2008. OCLC: ocm74524315.

[21] The American Heritage dictionary. Robot, 2023.

[22] KENNETH Y GOLDBERG. A brief history of robotics. In *Frontiers of Engineering: Reports on Leading Edge Engineering From the 1998 NAE Symposium on Frontiers of Engineering*, page 87. National Academies Press, 1999.

[23] Johanna Wallén. *The history of the industrial robot*. Linköping University Electronic Press, 2008.

[24] Dan Zhang. *Parallel robotic machine tools*. Springer, New York, 2010. OCLC: ocn428029615.

[25] Amouri Amar. *MODELISATION DYNAMIQUE D'UN ROBOT PARALLELE FORME DE PLUSIEURS MODULES EMPILES*. PhD thesis, University of Larbi Ben M'hidi Oum El Bouaghi, 2011.

[26] Ilian Bonev. Delta parallel robot-the story of success. *Newsletter, available at http://www. parallelmic. org*, 2001.

[27] Flavien PACCOT. *Contributions à la commande dynamique référencée capteur de robots parallèles*. PhD thesis, Université Blaise PASCAL – Clermont II Ecole Doctorale Science Pour l'Ingénieur de Clermont-Ferran, 2009.

[28] Taghirad, Hamid D. *Parallel Robots Mechanics and Control*. CRC Pr I Llc, 2017. OCLC: 982649253.

[29] Gerald Cook and Feitian Zhang. *Mobile robots: Navigation, control and sensing, surface robots and AUVs*. John Wiley & Sons, 2020.

[30] Kevin M. Lynch, Nicholas Marchuk, and Matthew L. Elwin. Chapter 21 - sensors. In Kevin M. Lynch, Nicholas Marchuk, and Matthew L. Elwin, editors, *Embedded Computing and Mechatronics with the PIC32*, pages 317–340. Newnes, Oxford, 2016.

[31] Stefan Staicu. *Dynamics of Parallel Robots*. Springer, 2019.

[32] Saeed B. Niku. *Introduction to robotics: analysis, control, applications*. Wiley, Hoboken, N.J, 2nd ed edition, 2011.

[33] Prof. Alessandro De Luca. Robot components: Proprioceptive sensors. http://www.diag.uniroma1.it/deluca/rob1_en/04_CompsSensorsProprio.pdf/, 2014.

[34] Sebastien Briot and Wisama Khalil. *Dynamics of Parallel Robots: from rigid bodies to flexible elements*. Springer International Publishing, Switzerland, 2015. OCLC: 1042214746.

[35] Lucas Campos, Francis Bourbonnais, Ilian A. Bonev, and Pascal Bigras. Development of a Five-Bar Parallel Robot With Large Workspace. In *Volume 2: 34th Annual Mechanisms and Robotics Conference, Parts A and B*, pages 917–922, Montreal, Quebec, Canada, January 2010. ASMEDC.

[36] J. Lenarčč#x8D; and F. Thomas. *Advances in Robot Kinematics*. Springer Netherlands, Dordrecht, 2010. OCLC: 961058806.

[37] Coralie Germain. *Conception d'un robot parallèle à deux degrés de liberté pour des opérations de prise et de dépose*. PhD thesis, Ecole Centrale de Nantes (ECN), 2013.

[38] Richard G Cobb, Jeanne M Sullivan, Alok Das, L Porter Davis, T Tupper Hyde, Torey Davis, Zahidul H Rahman, and John T Spanos. Vibration isolation and suppression system for precision payloads in space. *Smart Materials and Structures*, 8(6):798, 1999.

[39] Adept Quattro, un robot industriel ultrarapide. https://www.zonerobotique.com/adept-quattro-un-robot-industriel-ultrarapide/, 2017.

[40] Minglei Zhu, Cong Huang, Shijie Song, and Dawei Gong. Design of a gough–stewart platform based on visual servoing controller. *Sensors*, 22(7):2523, 2022.

[41] Kevin G Gim, Joohyung Kim, and Katsu Yamane. Design and fabrication of a bipedal robot using serial-parallel hybrid leg mechanism. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5095–5100. IEEE, 2018.

[42] Sen Qian, Bin Zi, Wei-Wei Shang, and Qing-Song Xu. A Review on Cable-driven Parallel Robots. *Chin. J. Mech. Eng.*, 31(1):66, December 2018.

[43] Ilija Vukorep. Autonomous big-scale additive manufacturing using cable-driven robots. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 34. IAARC Publications, 2017.

[44] Stefan Staicu and Stefan Staicu. Spatial parallel robots. *Dynamics of Parallel Robots*, pages 191–243, 2019.

[45] Maximilian Lesellier, Loic Cuvillon, Jacques Gangloff, and Marc Gouttefarde. An active stabilizer for cable-driven parallel robot vibration damping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5063–5070. IEEE, 2018.

[46] G Brandt, Klaus Radermacher, Stéphane Lavallée, H W Staudte, and Günther Rau. A compact robot for image guided orthopedic surgery: Concept and preliminary results. In *CVRMed-MRCAS'97: First Joint Conference Computer Vision, Virtual Reality and Robotics in Medicine and Medical Robotics and Computer-Assisted Surgery Grenoble, France, March 19–22, 1997 Proceedings*, pages 767–776. Springer, 1997.

[47] Kenneth W Grace, J Edward Colgate, Matthew R Glucksberg, and John H Chun. A six degree of freedom micromanipulator for ophthalmic surgery. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 630–635. IEEE, 1993.

[48] Moshe Shoham, Michael Burman, Eli Zehavi, Leo Joskowicz, Eduard Batkilin, and Yigal Kunicher. Bone-mounted miniature robot for surgical procedures: Concept and clinical applications. *IEEE Transactions on Robotics and Automation*, 19(5):893–901, 2003.

[49] A Wolf, B Jaramaz, B Lisien, and AM DiGioia. Mbars: mini bone-attached robotic system for joint arthroplasty. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 1(2):101–121, 2005.

[50] Jiao Mo, Zhu-Feng Shao, Liwen Guan, Fugui Xie, and Xiaoqiang Tang. Dynamic performance analysis of the x4 high-speed pick-and-place parallel robot. *Robotics and Computer-Integrated Manufacturing*, 46:48–57, 2017.

[51] Rory Smith. ABB launches five axes Delta robot fastest for lightweight product picking, packing & re-orientation. https://new.abb.com/news/detail/93173/prsrl-ABB-launches-five-axis-Delta-robot-fastest-for-lightweight-product-picking-packing-and-re 2022.

[52] W Gharieb and G Nagib. Fuzzy intervention in pid controller design. In *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*, volume 3, pages 1639–1643. IEEE, 2001.

[53] Anna Jadlovska, Štefan Jajčišin, and Richard Lonščák. Modelling and pid control design of nonlinear educational model ball & plate. In *17th International Conference on Process Control*, volume 2, pages 871–874, 2009.

[54] Manashita Borah, Prasanta Roy, and Binoy Krishna Roy. Enhanced performance in trajectory tracking of a ball and plate system using fractional order controller. *IETE Journal of Research*, 64(1):76–86, 2018.

[55] Chi-Cheng Cheng and Chen-Hsun Tsai. Visual servo control for balancing a ball-plate system. *International Journal of Mechanical Engineering and Robotics Research*, 5(1):28, 2016.

[56] David Núñez, Gustavo Acosta, and Jovani Jiménez. Control of a ball-and-plate system using a state-feedback controller. *Ingeniare. Revista chilena de ingeniería*, 28(1):6–15, 2020.

[57] Wei Huang, Yixin Zhao, Yuguang Ye, and Wenjing Xie. State feedback control for stabilization of the ball and plate system. In *2019 Chinese control conference (CCC)*, pages 687–690. IEEE, 2019.

[58] Zhankui Song and Kaibiao Sun. Adaptive backstepping sliding mode control with fuzzy monitoring strategy for a kind of mechanical system. *ISA transactions*, 53(1):125–133, 2014.

[59] N Yubazaki, Jianqiang Yi, M Otani, N Unemura, and K Hirota. Trajectory tracking control of unconstrained objects based on the sirms dynamically connected fuzzy inference model. In *Proceedings of 6th International Fuzzy Systems Conference*, volume 2, pages 609–614. IEEE, 1997.

[60] Hongrui Wang, Yantao Tian, Ce Ding, Qing Gu, and Feng Guo. Output regulation of the ball and plate system with a nonlinear velocity observer. In *2008 7th World Congress on Intelligent Control and Automation*, pages 2164–2169. IEEE, 2008.

[61] Ming Bai, Huiqiu Lu, Jintao Su, and Yantao Tian. Motion control of ball and plate system using supervisory fuzzy controller. In *2006 6th World Congress on Intelligent Control and Automation*, volume 2, pages 8127–8131. IEEE, 2006.

[62] Yongyut Pattanapong and Chirdpong Deelertpaiboon. Ball and plate position control based on fuzzy logic with adaptive integral control action. In *2013 IEEE International Conference on Mechatronics and Automation*, pages 1513–1517. IEEE, 2013.

[63] Kyongwon Han, Yantao Tian, Yongsu Kong, Jinsong Li, and Yinghui Zhang. Tracking control of ball and plate system using a improved pso on-line training pid neural network. In *2012 IEEE international conference on mechatronics and automation*, pages 2297–2302. IEEE, 2012.

[64] Dejun Liu, Yantao Tian, and Huida Duan. Ball and plate control system based on sliding mode control with uncertain items observe compensation. In *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 2, pages 216–221. IEEE, 2009.

[65] Heeseung Bang and Young Sam Lee. Implementation of a ball and plate control system using sliding mode control. *IEEE access*, 6:32401–32408, 2018.

[66] David Debono and Marvin Bugeja. Application of sliding mode control to the ball and plate problem. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 1, pages 412–419. IEEE, 2015.

[67] Hazem I Ali, Haider M Jassim, and Amjad F Hasan. Optimal nonlinear model reference controller design for ball and plate system. *Arabian journal for science and engineering*, 44(8):6757–6768, 2019.

[68] Matej Oravec and Anna Jadlovská. Model predictive control of a ball and plate laboratory model. In *2015 IEEE 13th international symposium on applied machine intelligence and informatics (SAMI)*, pages 165–170. IEEE, 2015.

[69] M Oravec. Control of mechatronical systems using .net applications. *Master thessis. Supervisor: doc. Ing. Anna Jadlovská, PhD. Košice: TU–FEI*, 2014.

[70] Shorya Awtar, C Bernard, N Boklund, A Master, D Ueda, and Kevin Craig. Mechatronic design of a ball-on-plate balancing system. *Mechatronics*, 12(2):217–228, 2002.

[71] M Ali Rastin, Erfan Talebzadeh, S Ali A Moosavian, and Mojtaba Alaeddin. Trajectory tracking and obstacle avoidance of a ball and plate system using fuzzy theory. In *2013 13th Iranian conference on fuzzy systems (IFSC)*, pages 1–5. IEEE, 2013.

[72] Nima Mohajerin, Mohammad B Menhaj, and Ali Doustmohammadi. A reinforcement learning fuzzy controller for the ball and plate system. In *International conference on fuzzy systems*, pages 1–8. IEEE, 2010.

[73] Firas Al-Haddad. Control of a ball and plate system using model-based controllers. Master's thesis, Palestine Polytechnic University-mechatronics, 2020.

[74] Gyroscope sensor working and its applications. https://www.elprocus.com/gyroscope-sensor/#:~:text=What%20is%20A%20Gyroscope%20Sensor,only%20measure%20the%20linear%20motion., 2023. Accessed: 2023-02-01.

[75] Hantouchusa how it works: 4-wire analog-resistive touch screens. https://www.sparkfun.com/datasheets/LCD/HOW%20DOES%20IT%20WORK.pdf, 2023. Accessed: 2023-02-02.

[76] Antonio Yarza, Victor Santibanez, and Javier Moreno-Valenzuela. Global asymptotic stability of the classical pid controller by considering saturation effects in industrial robots. *International Journal of Advanced Robotic Systems*, 8(4):36, 2011.

[77] Ernesto Fabregas, Sebastián Dormido-Canto, and Sebastián Dormido. Virtual and remote laboratory with the ball and plate system. *IFAC-PapersOnLine*, 50(1):9132–9137, 2017.

[78] Vinayak Garg, Astik Gupta, Amit Singh, Yash Jain, Aishwarya Singh, Shashanka Devrapalli, and Jagannath Mohan. Anticipatory postural adjustments for balance control of ball and beam system. In *Computational Signal Processing and Analysis*, pages 33–43. Springer, 2018.

[79] Ashraf Saleem, Rateb Issa, and Tarek Tutunji. Hardware-in-the-loop for on-line identification and control of three-phase squirrel cage induction motors. *Simulation Modelling Practice and Theory*, 18(3):277–290, 2010.

[80] techno science. Conception assistée par ordinateur, 2020.

[81] Introducing solidworks. https://files.solidworks.com/pdf/introsw.pdf, 2014. Accessed: 2023-02-05.

[82] IA Magomedov and ZS Sebaeva. Comparative study of finite element analysis software packages. In *Journal of Physics: Conference Series*, volume 1515, page 032073. IOP Publishing, 2020.

[83] Thomas Brand. Isolated spi communication made easy, 2019.

[84] Serial bus smart control servo scs15 manual. https://grobotronics.com/images/companies/1/datasheets/SCS15&SCS115%20Manual.pdf?1516269264467, 2016. Accessed: 2023-02-07.

[85] DYNAMIXEL PROTOCOL 1.0. https://emanual.robotis.com/docs/en/dxl/protocol1/, 2023.

[86] Communication protocol manual. https://us-icbuim-file.oss-us-east-1.aliyuncs.com/file/2e73a8eb556b610b1c3612626f19b12f.pdf?Expires=1676471724&OSSAccessKeyId=LTAIxlzvmECyZvmX&Signature=6tjp97Hy2ltabjTB6hnEhalBaNI%3D&response-content-disposition=fileName%3DSmart%20%20Bus%20Servo%20Communication%20Protocol%20Manual210923.pdf/, 2023.

[87] Espressif Systems. Esp32, 2023.

[88] Espressif Systems. Esp32 series datasheet, 2019.

[89] Feetech FE-URT-1 USB to TTL, 485 bus programmer. https://www.feetechrc.com/FE-URT1-C001.html/, 2020. Accessed: 2023-02-14.

[90] Dimitrios Serpanos and Tilman Wolf. *Architecture of network systems*. Elsevier, 2011.

[91] Tobias Bjerregaard and Shankar Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Surveys (CSUR)*, 38(1):1–es, 2006.

[92] Danial Jahed Armaghani and Panagiotis G Asteris. A comparative study of ann and anfis models for the prediction of cement-based mortar materials compressive strength. *Neural Computing and Applications*, 33(9):4501–4532, 2021.

[93] Hakan Basarir, Mohamed Elchalakani, and Ali Karrech. The prediction of ultimate pure bending moment of concrete-filled steel tubes by adaptive neuro-fuzzy inference system (anfis). *Neural Computing and Applications*, 31(2):1239–1252, 2019.

[94] Mark Dowson. The ariane 5 software failure. *ACM SIGSOFT Software Engineering Notes*, 22(2):84, 1997.

[95] Bashar Nuseibeh. Ariane 5: who dunnit? *IEEE Software*, 14(3):15, 1997.