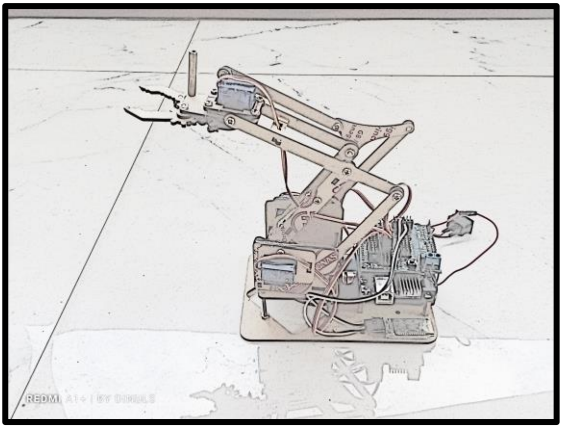
Research Book

Endeavour:

A Comprehensive Guide to Building a 4DOF Robotic Arm



W.V. Dinul Sasnada Imandiv

THURSTAN COLLEGE JUNIOR SCIENCE SOCIETY

DEDICATION

For the dear Thurstan College mother taking us from the darkness to the light…

SALUTATION

- To the principal, Mr. Pramuditha Wickramasinghe who gave us courage and opportunities.
- Our teacher-in-charge of the Junior Science Society of Thurstan College, Mr. Indika Navarathne who believes in me to give me this opportunity and helps with courage and advice.
- My research team-mates, Manitha Ishen Waduge, and Chanul Hansana, help me with their ideas, time, and resources.
- Three of my friends, Nileesha Thathsara, Vihitha Kenula, and Induwara Perera, help this research with their feedback, ideas, and thoughts.
- The members of the Thurstan College Junior Science Society, help me with their ideas.

PREFACE

"Endeavour: A Comprehensive Guide to Building a 4DOF Robotic Arm" is a research book designed to teach you the basics of robot arms, show you how to build this "Endeavour—4DOF Robot Arm" from scratch, and show you the journey of this research.

Building a robot is a huge milestone for anyone, from a kid to a mature adult, but the problem they have is that they have no idea about where to start, what parts to use, how to program a robot, and most importantly, how to troubleshoot a problem.

So, that's why this research book will be a help to you. Here, you'll be able to make your own "4DOF Robot Arm" in an easy, yet affordable manner which can be controlled by Bluetooth.

This research focuses on affordability, usability, and easy usage and maintenance giving you no hassle with this research, the main reason people abandon their ideas and projects. Also, you will get to learn some general troubleshooting steps that can be useful to save your projects money, effort, and mentality.

I hope that you'll have a good time with this research book which is made to be understandable for beginners and people with little to no experience with robotics to build your first robot and learn more about robotics, programming, and also, research.

Endeavour: A Comprehensive Guide to Building a 4DOF Robotic Arm

W.V. Dinul Sasnada Imandiv

"Endeavour" Researcher

01st November, 2024

CONTENTS

CHAPTER 1: INTRODUCTION TO ROBOTICS AND 4DOF SYSTEMS

WHAT IS ROBOTICS?

BRIEF HISTORY OF ROBOTICS

- The field of robotics has evolved dramatically over centuries, transforming from simple mechanical devices to advanced machines with complex functionalities. Here’s an overview of this journey:

EARLY MECHANICAL AUTOMATA

- Ancient Times: The concept of automata dates back to ancient civilizations. In Ancient Greece, engineers like the “Hero of Alexandria” designed self-operating machines. Hero's inventions, such as a mechanical bird that could "*sing*" and a steam-powered device called an “aeolipile”, were some of the earliest examples of machines designed to mimic life.
- Middle Ages: Inventors and artisans crafted elaborate clockwork mechanisms during the medieval period. For example, the “astronomical clocks” in European

cathedrals featured moving figurines and representations of celestial bodies, showing the intricate skills of the time.

- The Renaissance: Leonardo da Vinci sketched designs for a mechanical knight in 1495, which could sit, wave its arms, and even open its visor. Although it wasn’t fully constructed in his time, his ideas laid the groundwork for future humanoid machines.

INDUSTRIAL REVOLUTION (18TH-19TH CENTURIES)

- The Industrial Revolution marked a significant leap in machinery and automation. Factories employed “steam-powered machines” to perform repetitive tasks, especially in textile manufacturing.
- “Charles Babbage” and “Ada Lovelace” worked on the concept of a programmable machine, known as the Analytical Engine, in the mid-1800s. Though it was never completed, this invention paved the way for programmable systems and, ultimately, modern computers.

20TH CENTURY: RISE OF MODERN ROBOTICS

- 1940s-1950s: The development of computers after World War II allowed for programmable devices. The first true "robot" in the modern sense, “Unimate”, was developed in 1954 by George Devol. It was a programmable robotic arm, initially used in General Motors factories to perform repetitive, hazardous tasks, revolutionizing manufacturing processes.

- 1960s-1970s: The field of robotics continued to grow, with notable advancements like “Shakey the Robot” (1966), developed by Stanford Research Institute. Shakey was the first mobile robot capable of reasoning about its actions, marking a milestone in artificial intelligence (AI).
- 1980s-1990s: Robotics gained popularity beyond manufacturing. The invention of microprocessors and advancements in AI led to robots with greater autonomy and intelligence. “ASIMO”, a humanoid robot developed by Honda in 2000, showcased advances in mobility, speech recognition, and human-robot interaction.
- 21st Century: Robotics in Diverse Applications:
  Medicine: Robots like the “da Vinci Surgical System” have revolutionized healthcare, allowing surgeons to perform precise, minimally invasive procedures. Robots are also used in rehabilitation, prosthetics, and diagnostics.
  Manufacturing: Modern factories employ advanced robotic arms that work collaboratively with humans (cobots) and adapt to various tasks with high precision. Robotic process automation (RPA) also handles data-heavy tasks in the business sector.
  Space Exploration: Robots play a critical role in space exploration. For example, “NASA’s Mars rovers”, such as

Curiosity and Perseverance, are equipped with cameras, sensors, and drills to explore Mars's surface autonomously, collecting invaluable data and samples from a distant planet.
Daily Life: Robots are increasingly present in daily life, from household robots like “Roomba” vacuums to AI-driven digital assistants such as Alexa and Siri, which help with daily tasks and provide personalized assistance.

- This historical journey shows how robotics has evolved from simple machines to complex systems capable of operating in harsh, unpredictable environments, from factory floors to distant planets.

CORE CONCEPTS IN ROBOTICS

ACTUATION

- Definition: Actuation is the process that enables a robot to move and interact with its environment. Actuators are the components responsible for converting energy (usually electrical or pneumatic) into motion.
- Types: Common actuators include “motors” (electric motors, stepper motors, and Servos) and “hydraulic/pneumatic cylinders”. *In our 4DOF robotic arm, we use “Servo motors” to precisely control the arm’s movements.*

- Role in Robotics: Actuators provide the physical movement needed for a robot to carry out tasks. In robotic arms, actuators enable joints to move, allowing the arm to perform actions like lifting, rotating, and gripping objects.

SENSORS

- Definition: Sensors allow a robot to "sense" its surroundings by collecting data from the environment. This information helps the robot make decisions or adjust its actions accordingly.
- Types: There are various types of sensors in robotics:
  - Proximity Sensors (detect distance to nearby objects)
  - IR Sensors (detect objects using infrared light)
  - Gyroscopes and Accelerometers (measure orientation and movement)
  - Cameras (enable visual perception)
- Role in Robotics: Sensors enable a robot to respond to environmental changes. For example, a sensor can detect if an object is nearby and signal the robotic arm to stop or avoid the obstacle. This interaction between the robot and its environment is crucial for safe, efficient operations.

CONTROL SYSTEMS

- Definition: A control system is a mechanism that manages and directs the behavior of other components in a robot. It's the "brain" that takes inputs (from sensors) and translates them into outputs (through actuators).
- Types: Control systems can range from simple feedback loops to complex algorithms using artificial intelligence.
- Role in Robotics: Control systems enable autonomous decision-making. *For instance, in our robotic arm, the control system (programmed on an Arduino) receives commands from the Bluetooth app and adjusts the Servo positions based on this input, allowing the arm to follow instructions accurately.*

POWER SUPPLY

- Definition: Robots need a reliable power source to function. This could be a battery, solar power, or a direct power connection, depending on the robot's application.
- Types: Battery types range from "Li-Ion (Lithium-Ion)" for portable robots to "AC power" for stationary robots.
- Role in Robotics: A stable power supply is essential to ensure that all electronic components operate within safe voltage and current limits. *For the 4DOF robot arm, the power supply includes a "7.4V Li-Ion battery" and a*

capacitor for stabilization to prevent fluctuations that could damage components.

PROGRAMMING AND CONTROL LOGIC

- Definition: Programming is the language through which we "*speak*" to a robot, telling it how to respond to inputs and perform tasks.
- Programming Language: Arduino uses C/C++-based code, while the Bluetooth app uses Java and Kotlin and Android Studio.
- Role in Robotics: Through programming, the robot's actions become repeatable and precise. A well-programmed control system ensures that the robot arm moves smoothly and communicates with the Bluetooth app properly.

SUMMARY

- These core concepts—actuation, sensing, control systems, power supply, and programming—form the foundation of robotics. By understanding these elements, students and readers gain insight into how robots, including the 4DOF robotic arm, work and interact with their environment. This background knowledge will also provide context for more hands-on sections in the book, where readers can apply these principles to assemble and program their own robotic arm.

UNDERSTANDING DEGREES OF FREEDOM (DOF)

- In robotics, "Degrees of Freedom" (DOF) refers to the number of independent movements a robot or robotic arm can make. Each degree of freedom represents a way in which a robot can move, pivot, or rotate in space. The concept of DOF is fundamental to understanding how versatile and capable a robot is in performing complex tasks, especially in three-dimensional space.

DEFINITION OF DOF

- A single "degree of freedom" corresponds to one independent movement. For instance, moving up or down, rotating left or right, or extending and retracting are each considered individual degrees of freedom.
- In a robotic arm, each joint or axis typically represents one degree of freedom. The more joints a robotic arm has, the more degrees of freedom it possesses, allowing it to reach and manipulate objects in different positions.

TYPES OF MOTION

- In robotics, there are generally two main types of motion that contribute to a robot's degrees of freedom: Translational Motion: Linear movement along an axis, like moving up/down, left/right, or forward/backward. Rotational Motion: Rotational movement around an axis, such as rotating or pivoting at a joint. This includes rotation around the X, Y, or Z axes (known as roll, pitch, and yaw, respectively).

DOF IN ROBOTIC ARMS

- Robotic arms are often classified by their degrees of freedom, which indicate how many ways they can move and manipulate objects. For example:
  2-DOF Robotic Arm: Capable of moving in two distinct ways, such as rotating at the base and extending the arm up or down. Limited in versatility, it's often used for simple, repetitive tasks.
  3-DOF Robotic Arm: Can move along three different axes, allowing it to reach a wider range of positions and perform slightly more complex tasks.
  4-DOF Robotic Arm: *(like the one in the Endeavour project):* Has four independent movements. This setup often includes rotation at the base, an elbow joint for

bending, a wrist joint for further rotation, and sometimes a gripper for grasping objects. While 4DOF arms are limited in comparison to more advanced arms, they can perform many tasks requiring precision in confined spaces.
6-DOF Robotic Arm: A 6DOF arm has even more control over its movements, enabling it to reach almost any position and orientation in a 3D space. This is standard for industrial robots because it allows for high precision and flexibility in complex tasks.

IMPORTANCE OF DOF IN ROBOTIC APPLICATIONS

- Manipulation Flexibility: Higher DOF means a robot can reach around obstacles, access difficult angles, and adjust orientation to match the task's requirements. This is essential in fields like manufacturing, surgery, and space exploration, where precise movements are crucial.
- Complexity and Cost: More DOF increases a robot's versatility, but it also raises complexity in terms of control systems, programming, and cost. A balance must be struck between the number of degrees of freedom and the intended application.
- Task Suitability: For simpler tasks *(e.g., picking objects from a conveyor belt)*, a 2-DOF or 3-DOF robotic arm may be sufficient. However, for more intricate tasks like

assembling electronic components or performing surgery, a higher DOF robotic arm is required to handle delicate movements and orientations.

DOF IN ENDEAVOUR'S 4DOF ROBOTIC ARM

- In the Endeavour project, the 4DOF arm has four distinct movements, each carefully chosen to give the arm adequate functionality without over-complicating the design:
  Base Rotation: Allows the arm to pivot around a central point, giving it access to a circular workspace.
  Shoulder Joint: Provides an up-and-down motion, allowing the arm to reach higher or lower positions.
  Elbow Joint: Enables bending and extending of the arm, increasing its reach and making it possible to pick up objects from different distances.
  Gripper: Interacts with objects and grips the objects which are picked up and holds them in the manipulation process.
- By carefully selecting these four degrees of freedom, the Endeavour 4DOF robotic arm is capable of performing a wide range of tasks within a specific area, including grasping, moving, and orienting objects. This DOF setup makes it suitable for educational purposes and demonstrates core robotics principles without overwhelming students or hobbyists with excessive complexity.

PRACTICAL CONSIDERATIONS FOR DOF

- Programming Complexity: As the DOF increases, so does the complexity of programming the robot. Each DOF requires precise control, often involving intricate algorithms and sensor feedback to maintain accuracy.
- Mechanical Constraints: Physical limits and the strength of the joints restrict how much each DOF can be used. For example, adding more DOF without strengthening the chassis or structure can make a robot less stable.
- Energy Consumption: More DOF usually means more motors or actuators, which can drain power faster. This consideration is especially relevant for battery-powered robots, where energy efficiency is essential.

SUMMARY

- Degrees of Freedom (DOF) are a critical concept in robotics, defining a robot's range of movement and capability. For general readers and students, understanding DOF provides insight into how robots move and interact with their environment. In the Endeavour 4DOF robot arm project, each degree of freedom is optimized to balance functionality and simplicity, making it a practical example of robotics principles that can inspire further exploration into this field. This background knowledge equips readers to appreciate both the mechanics of robotic arms and the complexity of creating machines that can perform precise, coordinated tasks autonomously.

WHY 4DOF?

- In the Endeavour project, it uses a 4DOF geometry, a perfect balance in affordability, complexity, practicality, efficiency and usability.
- Because of this 4DOF geometry, it is capable of performing a wide range of tasks within a specific area, including grasping, moving, and orienting objects according to the commands from the Bluetooth app.
- Also, this makes it suitable for educational purposes and demonstrates core robotics principles without overwhelming students or hobbyists with excessive complexity.

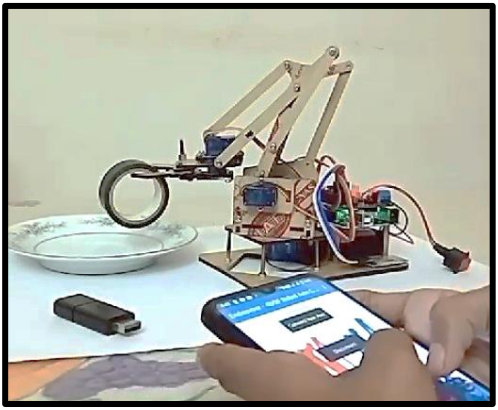


Figure 1.4.1: The Endeavour-4DOF Robot Arm lifts a roll of tape and manipulates its location.

OBJECTIVES OF THIS RESEARCH

OBJECTIVES:

- Prototype Development: Design and assemble a 4DOF robotic arm with a lifting capacity of 55 grams using an acrylic chassis and Tower Pro SG90 Servos.
- Safety Mechanisms: Incorporate features like a kill switch and a hardcoded Servo movement range limits for arm's longevity and flawless functioning.
- Bluetooth-Based Control: Implement an Android app to control the robot in real-time via Bluetooth (HC-05).

YOU'LL LEARN:

- To assemble a *4DOF Robot Arm* Chassis

- To program the *Robot Arm* you built
- To use the *Endeavour – 4DOF Robot Arm Controller* app
- To troubleshoot basic problems of the arm

OVERVIEW

- The Endeavour - 4DOF Robot Arm is a small-scale robotic arm designed to demonstrate precise control using affordable, commercially available components. It aims to perform basic tasks while integrating sensor-based interaction with its environment. Controlled via an Android app and Bluetooth, the project showcases how low-cost hardware can be used to build functional robotic systems.
- The Endeavour project focuses on designing a 4 Degree of Freedom (4DOF) robotic arm using commercially available components like Tower Pro SG90 Servos and an Arduino Uno microcontroller, housed in an acrylic chassis. The arm, while limited in lifting capacity, provides a functional platform for small-scale experimentation, allowing further research in robotic control systems.

CHAPTER 2: COMPONENTS OF THE 4DOF ROBOT ARM

ARDUINO MICROCONTROLLER

- The Endeavour Project utilizes an Arduino Uno, renowned for its simplicity, intuitiveness, programmability and practicality for affordable robotics with its feature-richness and stability.
- It also uses an Arduino Sensor Shield V5 to control Servos, optimizing power distribution and connections, increasing the arm's reliability and stability.

SPECIFICATIONS

Microcontroller	ATMega328P
Clock Speed	16MHz @ 5V
Digital Pins	14
Analog Pins	6
Programmed from	AvrDude
USB Port	USB type B
Voltage Regulator	AMS1117@ 5V

USB-to-TTL Converter	ATMega16U2 or WCH CH340G
Storage (Flash)	32KB
SRAM	2048B
EEPROM	1024B

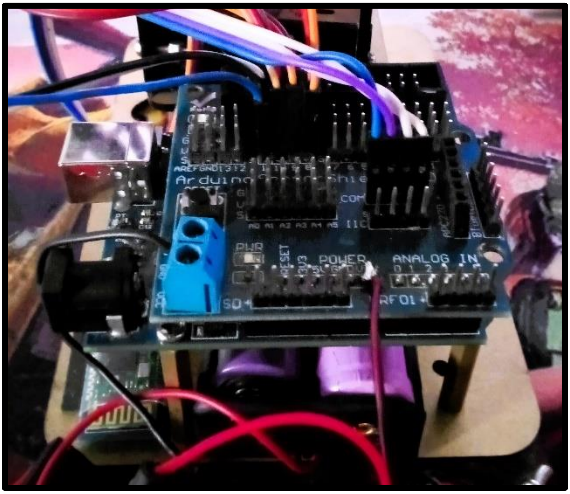


Figure 2.1.1: The Arduino Uno Board, the Sensor Shield and its connections to the arm's Servos and the HC-05 Bluetooth module.

WHY ARDUINO UNO?

- The Arduino Uno is a great starter board for beginners and novices in robotics because of the features and practicality for projects while being very affordable, a main focus of the Endeavour project.
- This Arduino Board also supports shields such as the Arduino Sensor Shield V5, which is used to make easy

connections with other components more easily and efficiently.

- The prescence of a good voltage regulator which can supply enough power for the Servos reliably without using a discrete voltage regulator, keeping the circuitry simple and inexpensive.

BLUETOOTH COMMUNICATION

- The HC-05 Bluetooth module is an easy-to-use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup which uses Bluetooth 2.0 + EDR (Extended Data Rate) to communicate with the Bluetooth app, Endeavour – 4DOF Robot Arm Controller as the slave of the connection.
- The similar HC-06 Bluetooth module can also be used as the only difference between the HC-05 and the HC-06 is that the HC-06 is slave only while the HC-05 supports both master and slave modes.

SPECIFICATIONS

Bluetooth Version	2.0 + EDR
Baud Rate	9600 (default), 19200, 38400, 57600, 115200 baud
Operating Voltage	3.6 – 6V (5V Recommended)

Speed	Speed: Asynchronous: 2.1 Mbps (Max) / 160 Kbps, Synchronous: 1Mbps/1Mbps
Passkey	1234
Mode	Interchangeable master/slave mode
Frequency	2.4GHz ISM band
Current	30 mA

WHY HC-05?

- The HC-05 Bluetooth module is more inexpensive compared to other Bluetooth modules on the market and still is sufficient for this project.
- The interchangeable master/slave mode will come in handy when you are going to improve the app to receive information from the Arduino because the HC-06 is a slave-only device.

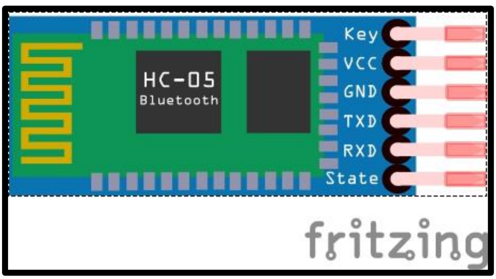


Figure 2.2.1: An image of a HC-05 (6 pin) Bluetooth module on Fritzing.

SERVO MOTORS

- SG90 Servos are cheap, easy-to-use, analog Servo motors that have a 30:1 gear ratio and they run off of PWM (Pulse Width Modulation) pulses, giving a ton of torque for a small robot arm such as the Endeavour robot arm.
- These Servos can be easily used with the *"Servo"* library, which is a very easy-to-use Arduino library that supports many Arduino boards and comes integrated with Arduino IDE.
- But it's recommended to use Tower Pro MG90S Servos if possible because the SG90 Servos have plastic gears while MG90S Servos have metal gears, which will help the arm lift heavier items without breaking gears inside.
- Even though it is recommended, you can still use SG90 Servos if you have a tighter budget and/or you don't mind the reduced lifting capacity. Just keep a close eye on the Servos for a longer lifetime.

Figure 2.3.1: Broken gears of a SG90 Servo motor after lifting 90 grams.

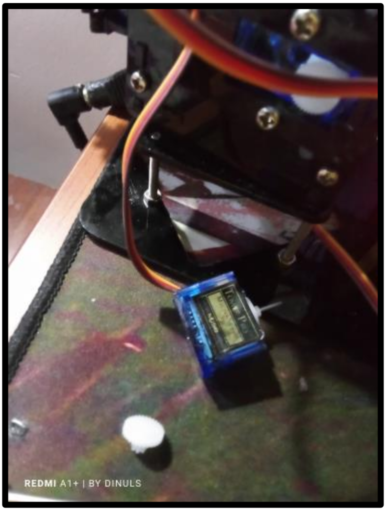


Figure 2.3.2: The broken SG90 Servo motor disassembled. The Servo was on the chassis C-001 which got replaced by the C-002 (present chassis) later on.

SPECIFICATIONS

Servo Name	Tower Pro™ SG90
Category	9g Micro Servo
Voltage	4.8 – 6V (5V Recommended)
Torque	1.5kg /cm @ 4.8V or 1.6kg/cm @ 6V
Pins	3 (GND, VCC, SIGNAL)
Current	10mA (idle) 150-250mA (moving) 360mA (stall)
Signal mode	PWM (Pulse Width Modulation)
Range of Degrees	0-180 (360 on some Servos)
Weight	9g

Dimensions (length, breadth, height)	22.2 x 11.8 x 31 mm (approximately)
Speed	0.1 s/60 degree
Temperature Range	0 ºC - 55 ºC
Idle PWM Pulse	50Hz
Gear Ratio	30:1

WHY SG90 SERVOS?

- Tower Pro SG90 Servos can be bought for a very low price and they're a good pick for hobbyists and smaller robots such as the Endeavour as a starting point. Even though I would personally recommend buying M90S Servos, this is still enough for the arm for basic functioning.
- These Servos can be controlled quite easily with the *"Servo"* library, giving amateur robotists little to no hassle in controlling the Servos.

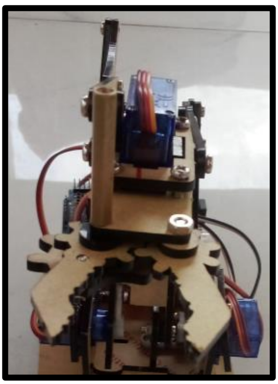


Figure 2.3.3: A Tower Pro SG90 Servo on the Endeavour's gripper used to hold objects.

POWER SUPPLY AND MANAGEMENT

- Every single electronic component needs a reliable source of power to function properly and reliably. The Endeavour, a portable robot arm, uses a 7.4V 7200mAh 2S Li-Ion (Lithium-ion) battery as the main power source.
- The arm also has a 3300µf capacitor to regulate sudden voltage spikes which can cause erratic behavior, sudden resets, or even board or Servo damage, helping the reliability of the arm in the long term.
- A stable voltage is required for the arm to function properly and the Li-Ion battery can reliably power the arm up to 2.5 hours after a full charge. The Li-Ion batteries are 2 cylindrical 18650 cells, connected in series.

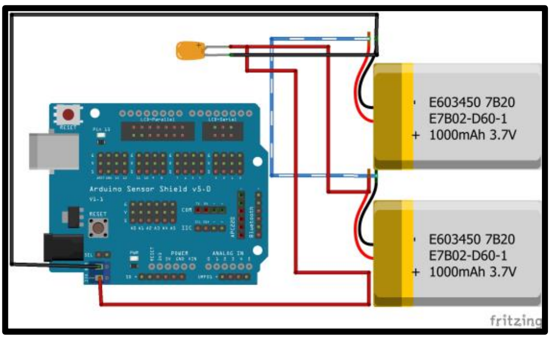


Figure 2.4.1: The Endeavour's power circuitry drawn on fritzing. (The 2 Li-Po cells represent the batteries).

WHY A CAPACITOR?

- Capacitors are used to stabilize the electricity and they are used to prevent sudden voltage spikes and drops, giving the arm to function properly with various current loads the added capacitor can help to prevent damage to the electronics done by voltage spikes.

CHASSIS AND STRUCTURAL MATERIALS

- The chassis uses acrylic as the main material and this table will take you across the pros and cons of acrylic:

Pros	Cons
Strength and Durability • Acrylic is impact-resistant and rigid, providing good structural support for lightweight applications.	Weight Limitations • While acrylic has good strength, it is not ideal for supporting heavy loads. In the Endeavour - 4DOF Robot Arm, the acrylic chassis limits the lifting capacity to about 55 grams.

Lightweight - Acrylic is significantly lighter than glass or metal, which helps minimize the overall weight of the robotic arm.	Fragility Under Stress - Acrylic can crack or chip if subjected to high impact or stress.
Aesthetic and UV Resistance - Black acrylic offers an appealing, polished look and resists UV exposure, helping maintain its color and structural integrity when exposed to light.	Requires Caution - To prevent cracking, always drill pilot holes before inserting screws or bolts. If using adhesive, apply only as much as needed to avoid stress.
Ease of Workability - Acrylic is easy to cut, drill, and shape with standard tools, making it ideal for creating custom chassis designs.	Surface Protection - Acrylic scratches more easily than harder materials.
Affordability - Acrylic is cost-effective and readily available in different thicknesses, making it a practical choice for educational and prototype robotic projects.	Chemical Resistance - Acrylic can be damaged by organic solvents like acetone or toluene, which may cause clouding, weakening, or cracking.

SUMMARY

- Acrylic’s blend of strength, affordability, and workability makes it a practical choice for the Endeavour 4DOF robot arm. While it’s suitable for light-duty structural applications, users should be mindful of its limitations under high stress and impact. Proper handling and thoughtful design can optimize acrylic’s strengths, making it ideal for robotic projects that balance functionality and aesthetics.

SUMMARY

- This chapter introduces each key component of the Endeavour 4DOF robot arm and explains its role in enabling movement, control, and stability:
  Arduino Uno Microcontroller: Acts as the "brain" of the robot, processing commands and controlling the arm’s movements. The Arduino's simplicity makes it suitable for beginners and allows for flexible programming.
  HC-05 Bluetooth Module: Enables wireless control of the robot via a Bluetooth connection with an Android app. This setup provides real-time interaction, allowing users to control the arm remotely.
  Tower Pro SG90 Servos: These small yet powerful motors control each joint, enabling precise movement at each degree of freedom. The servos are limited in

torque, making them ideal for lightweight applications like this project.
7.4V Li-Ion Battery with Power Stabilization: Powers the arm, with a capacitor added to stabilize the supply and prevent power surges that could damage components or disrupt smooth operation.
Black Acrylic Chassis: Provides a lightweight yet strong structure to hold all components together. The black acrylic material offers durability and an aesthetic appeal, though it limits the robot's weight capacity to light objects.

- Together, these components create a balanced system for precise control and stability, allowing readers to understand the core hardware and design decisions that shape the robot's functionality.

CHAPTER 3: BUILDING THE 4DOF ROBOT ARM

ASSEMBLING TUTORIAL

TOOLS NEEDED

- Small screwdriver
- Allen wrench (if using hex bolts)
- Small pliers
- Hot glue gun or appropriate acrylic adhesive (optional for stability)

STEP 1: PREPARE THE COMPONENTS

- Black acrylic pieces for the base, joints, and end effector
- Four Tower Pro SG90 Servos (for base rotation, shoulder, elbow, and gripper)
- Screws and nuts (usually provided with the Servos)
- Arduino Uno, HC-05 Bluetooth module
- Battery pack (7.4V Li-Ion) and capacitor for power stabilization

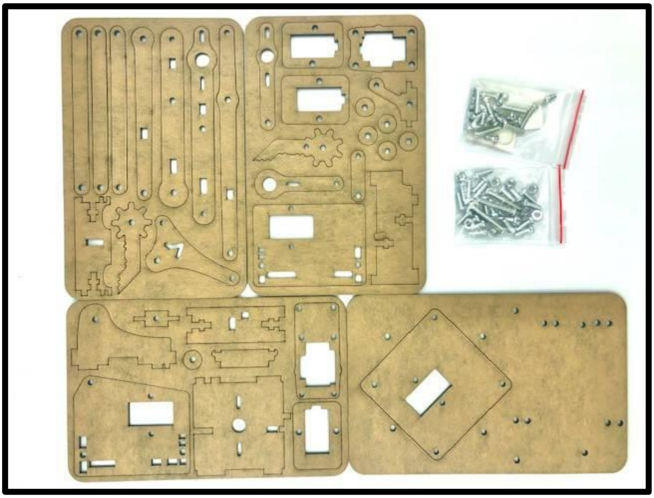


Figure 3.1.1: The acrylic parts with the protective cover which can be removed and the screws and bolts.

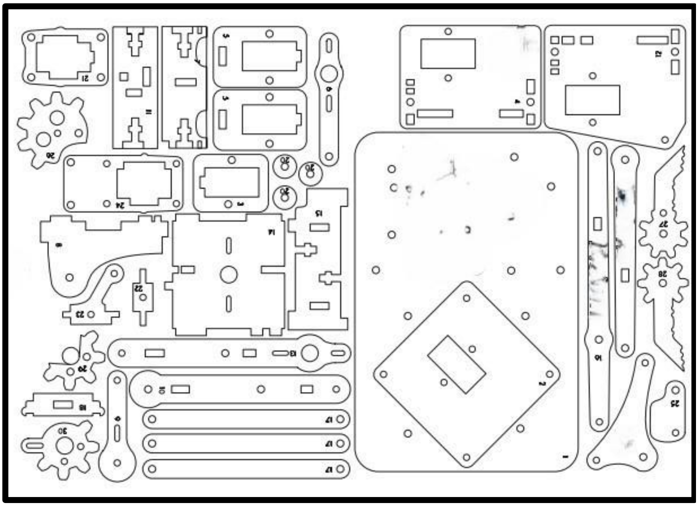
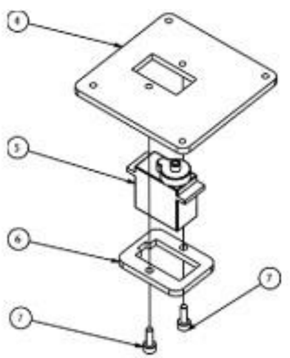


Figure 3.1.2: The acrylic parts, sourced from the ".dxf" file, numbered. This numbering will be helpful in the next steps.

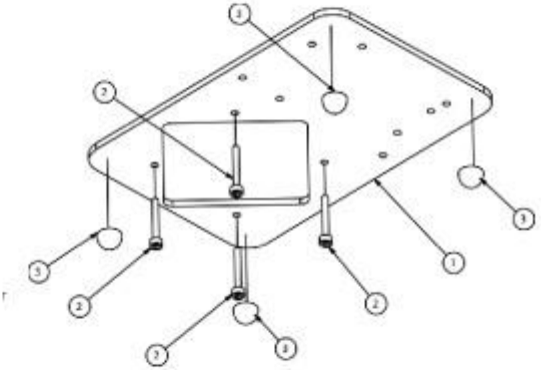
STEP 2: BASE ASSEMBLY

- Attach the Base Servo:
  Place the base servo (Servo 1) on the bottom acrylic piece designated for the base. Secure it with screws, ensuring the servo horn faces outward.
- Connect the Rotating Base Platform:
  Attach the rotating platform on top of the base servo horn, aligning it to allow smooth 180° rotation. Use screws to secure the horn in place, ensuring it's firmly attached.
- Follow the steps in the pictures:

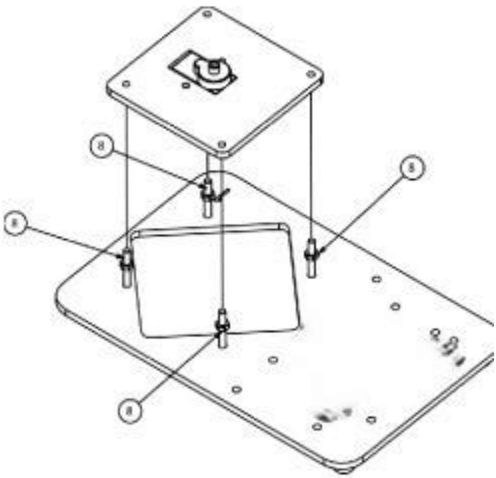
ATTACH THE BASE SERVO



1.



2.



3.

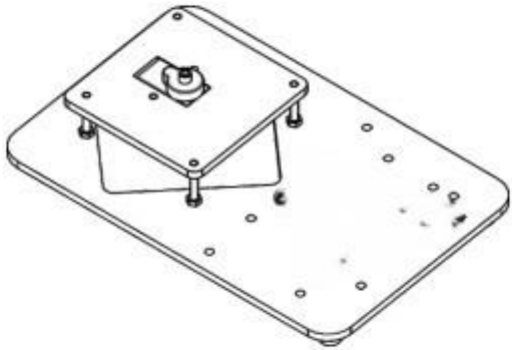
4. 

Figure 3.1.2.1, 3.1.2.2, 3.1.2.3., 3.1.2.4.: Steps to assemble the base of the arm respectively.

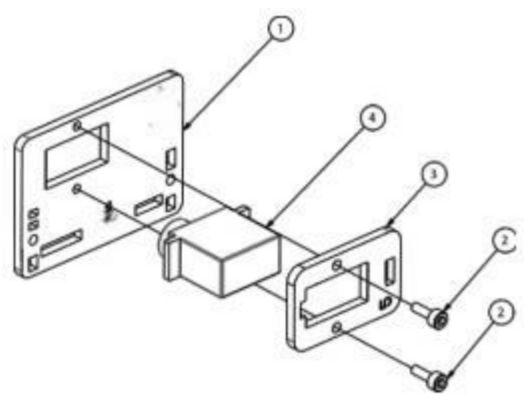
Parts List			
Item	Part Name	Quantity	Part No.
1	Base Plate	1	1
2	M3X20 Screw	4	
3	Pivot Servo Plate	1	2
4	Rubber stops	4	
5	SG90 Servo	1	
6	Base Servo Collar	1	3
7	M3X8 Screw	2	

8	M3 Nut	4	

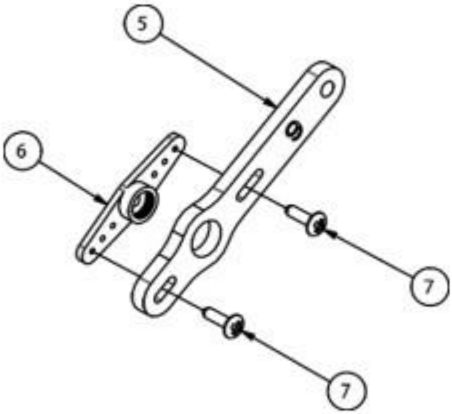
STEP 3: SHOULDER ASSEMBLY

- Attach the Shoulder Servo (Servo 2): Mount the shoulder servo vertically on the rotating platform. The servo horn should point upward, ready to connect with the shoulder piece.
- Install the Shoulder Joint: Attach the acrylic shoulder piece to the shoulder servo horn with screws. Make sure the shoulder piece moves freely without obstruction from the base.
- Tip: Tighten screws to prevent wobbles but check for smooth shoulder movement.
- Follow the steps in the pictures:

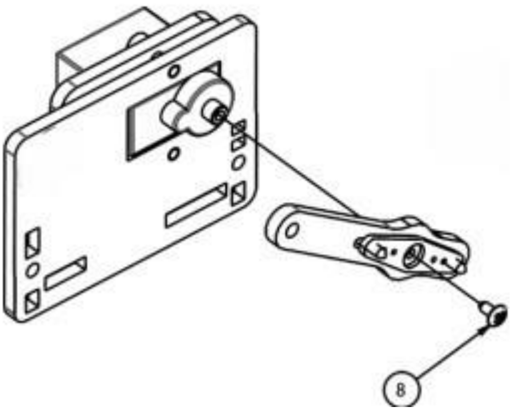
ATTACH THE SHOULDER SERVO



1.



2.



3.

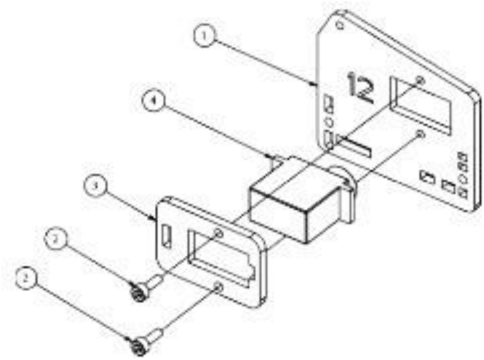
Figure 3.1.3.1, 3.1.3.2, 3.1.3.3.: Steps to assemble the shoulder of the arm respectively.

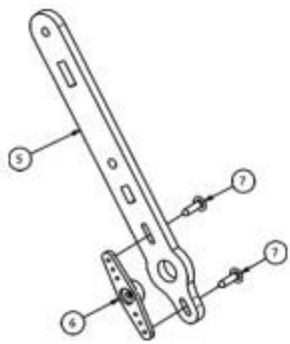
Item	Part Name	Quantity	Part No.
1	Shoulder Servo Plate	1	4
2	M3X8 Screw	2	
3	Servo Collar	1	5
4	SG90 Servo	1	
5	Long Servo Arm	1	6
6	Servo Double Arm	1	
7	Servo Mount Screw	2	
8	Servo Screw	1	

STEP 4: ELBOW ASSEMBLY

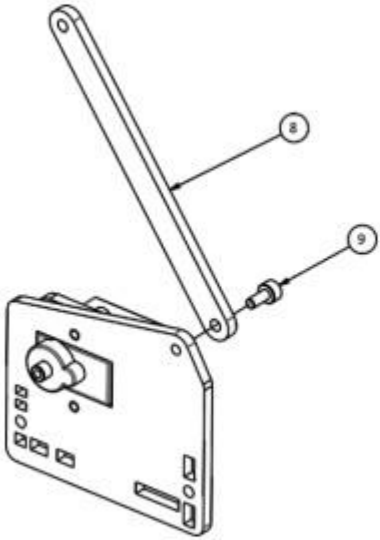
- Mount the Elbow Servo (Servo 3): Attach the elbow servo to the shoulder piece in a way that aligns it with the elbow joint. This setup allows the servo to support bending motions.
- Connect the Elbow Joint: Attach the elbow acrylic piece to the elbow servo horn and secure it with screws. Check for free movement and adjust if necessary.
- Follow the steps in the pictures:

ATTACH THE ELBOW SERVO

1. 

2. 

- Tip: Tighten screws to prevent wobbles but check for smooth shoulder movement.
- Tip: Never use an adhesive instead of screws because adhesives may damage the Servos and might make the chassis more fragile.

3. 

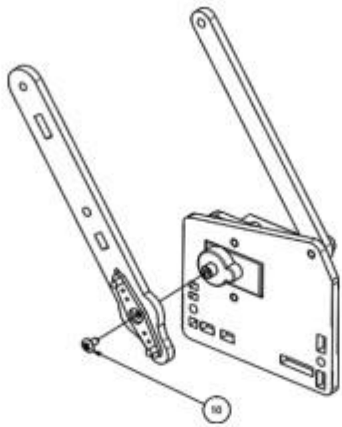
4. 

Figure 3.1.4.1, 3.1.4.2, 3.1.4.3.,3.1.4.4.: Steps to assemble the elbow of the arm respectively.

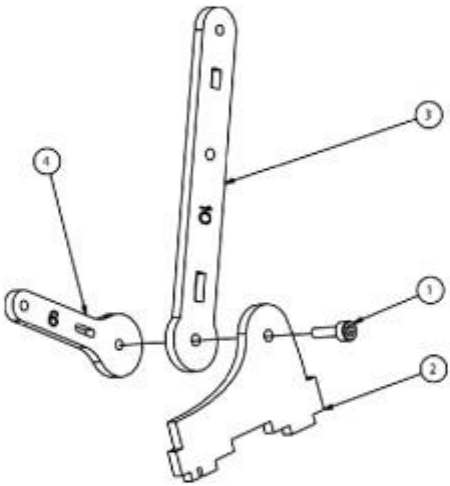
Item	Part Name	Quantity	Part No.
1	Elbow Servo Plate	1	12
2	M3X8 Screw	2	
3	Servo Collar	1	5
4	SG90 Servo	1	
5	Elbow Base Joint	1	13
6	Servo Double Arm	1	
7	Servo Mount Screw	2	
8	Parallel Linkage	1	17
9	M3X6 Screw	1	
10	Servo Screw	1	

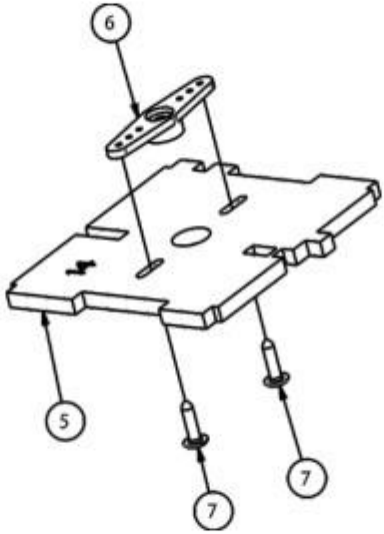
STEP 5: CARRIER ASSEMBLY

- Tip: After securing the main arm cross web and tightening screws from previous steps, gently test the movement of the parallel linkage arms. Ensure they move freely without any excessive friction.

- Tip: Use a screwdriver to tighten the screws, but be cautious not to over-tighten. This could cause the acrylic to crack or add unwanted pressure, making the linkage hard to move.
- This is the longest step in the assembly and this step must be done precisely for the arm's proper functioning.
- Assemble the carrier following the pictures:

ASSEMBLE THE BASE PLATE

1. 



2.

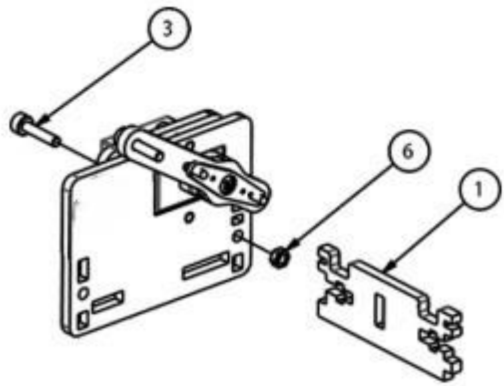
Figure 3.1.5.1, 3.1.5.2.: First steps to assemble the carrier of the arm respectively.

Item	Part Name	Quantity	Part No.
1	M3X10 Screw	1	
2	Shoulder Mount Tablet	1	8
3	Shoulder Base Joint	1	10
4	Short Servo Arm	1	9

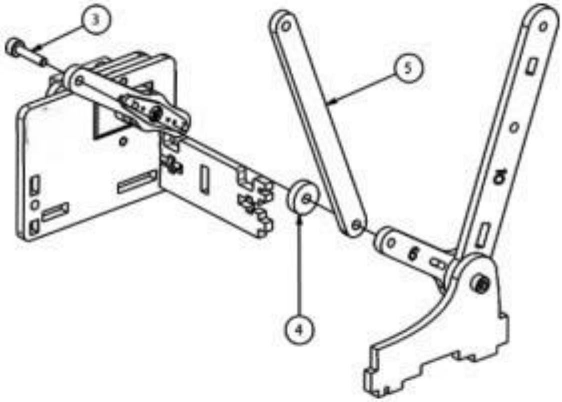
5	Arm Bottom Plate	1	14
6	Servo Double Arm	1	
7	Servo Mount Screw	2	

- Tip: A small amount of silicone-based lubricant at the joints can improve movement smoothness, but avoid oil-based lubricants as they may damage acrylic over time.
- Tip: As you progress with assembly, ensure cables from servos and sensors are neatly routed to avoid getting tangled or caught in moving parts.
- You have to do some additional steps to follow the below pictures:

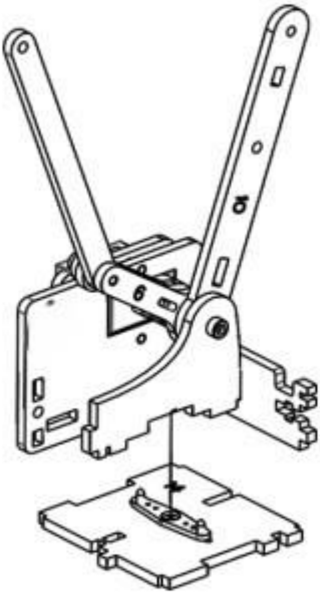
ASSEMBLE THE BASE CARRIER



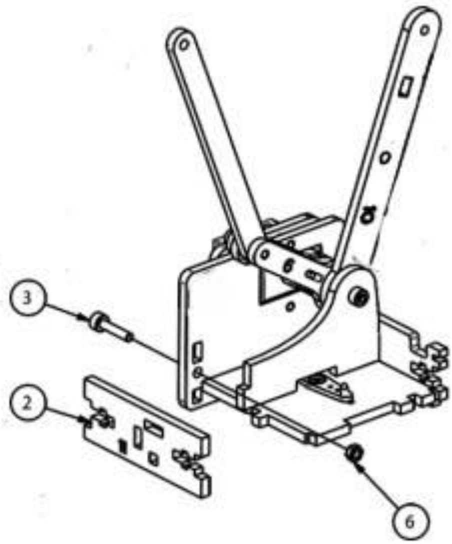
1.



2.



3.



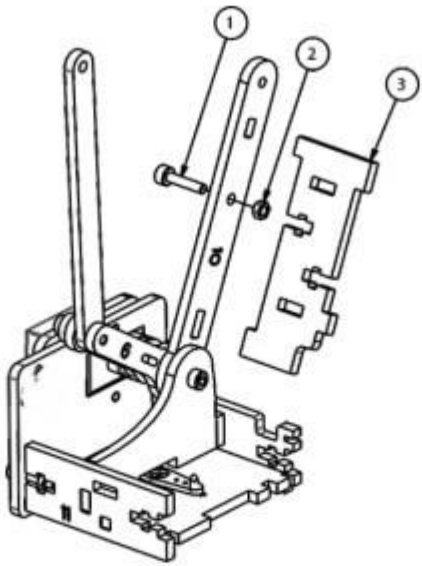
4.

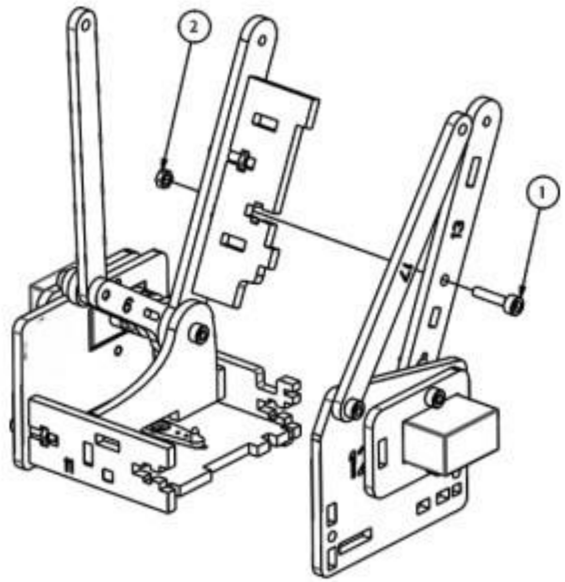
Figure 3.1.5.3, 3.1.5.4, 3.1.5.6.,3.1.5.7.: Steps to assemble the carrier of the arm respectively (2nd phase).

Item	Part Name	Quantity	Part No.
1	Front Arm Base Cross	1	7
2	Back Arm Base Cross	1	11
3	M3X12	3	
4	Spacer	1	20
5	Parallel Linkage	1	17
6	M3 Nut	2	

- Tip: After assembling, adjust the screws for a snug fit, allowing the carrier to function as intended.
- To finish the carrier, do the final step as the pictures show:

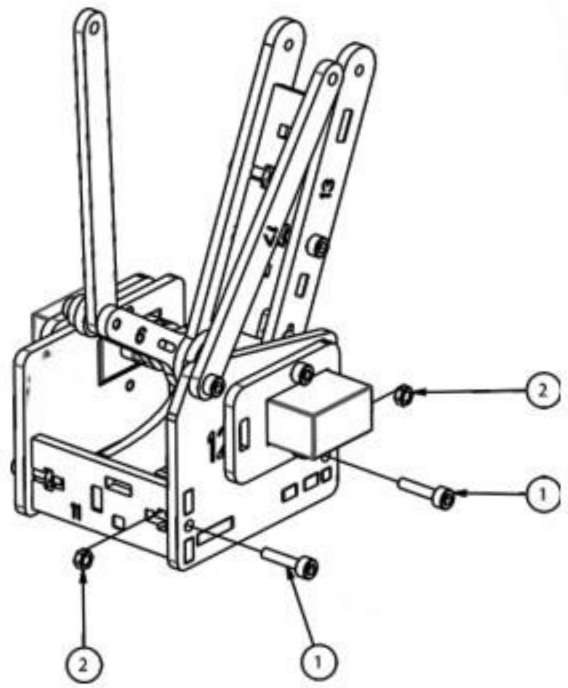
ASSEMBLE THE WRIST CARRIER

1. 



2.

- Tip: Once everything is secure, test the arm's range of motion. Move it through its full DOF to check for any binding or resistance. If you encounter resistance, slightly loosen and adjust as necessary.



3.

Figure 3.1.5.8, 3.1.5.9, 3.1.5.10.: Steps to assemble the carrier of the arm respectively (final phase).

Item	Part Name	Quantity	Part No.
1	Servo Screw	1	

2	Left Wrist Joint	1	16
3	M3X6 Screw	2	

- After assembling, the parts should look similar:

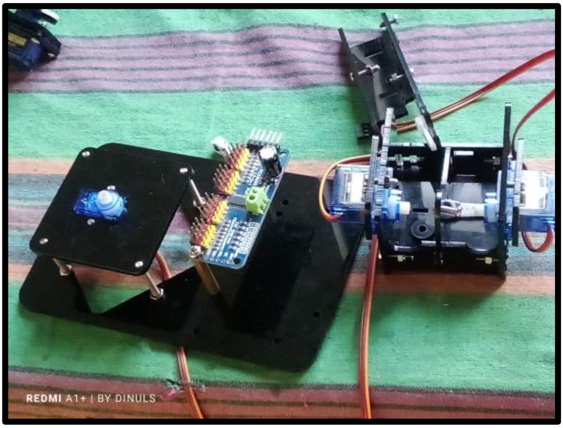


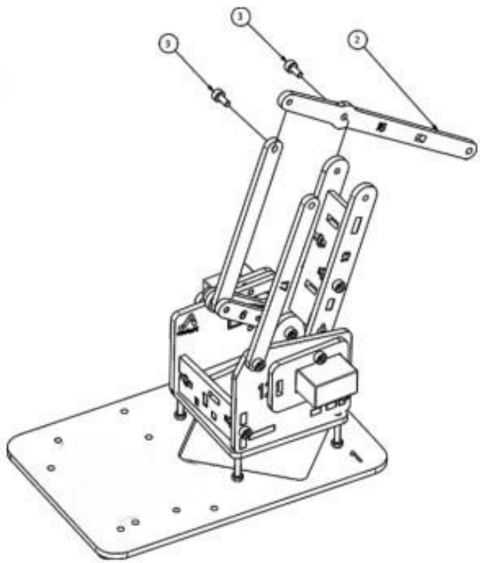
Figure 3.1.5.11.: The base and carrier assembled, but not the linkages in the chassis C-001.

STEP 6: WRIST AND GRIPPER ASSEMBLY

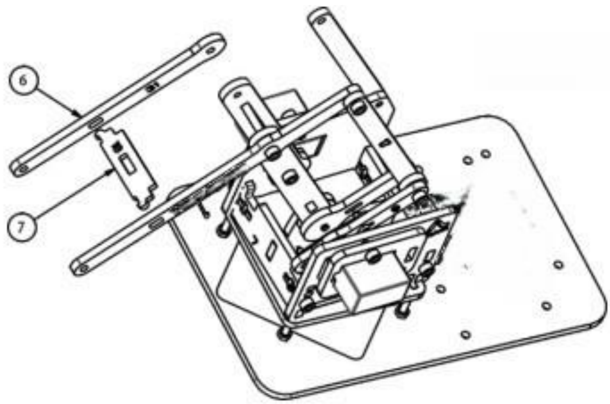
- Attach the Gripper Mechanism: If using a gripper, attach it to the wrist joint with screws. Ensure the gripper opens and closes without interference.
- Tip: Test the range to verify smooth grip and release actions.

- The wrist and the gripper are assembled following the pictures:

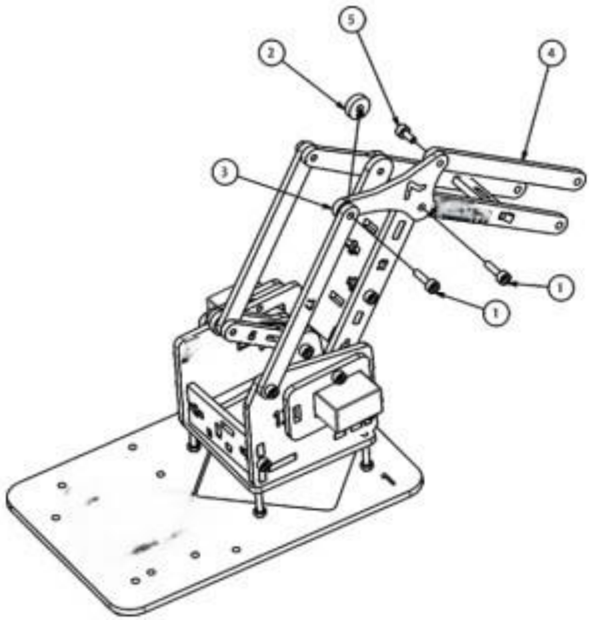
ASSEMBLE THE WRIST



1.



2.



3.

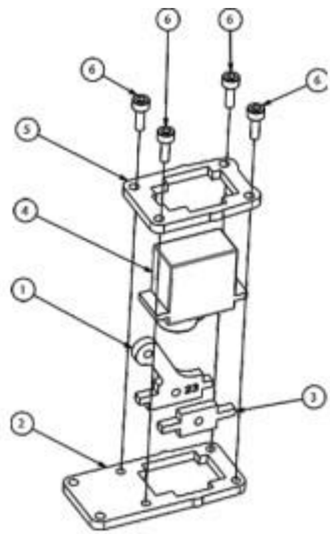
Figure 3.1.6.1, 3.1.6.2, 3.1.6.2.: Steps to assemble the wrist of the arm respectively.

Item	Part Name	Quantity	Part No.
1	M3X10 Screw	2	
2	Spacer	1	20
3	Parallel Linkage Connector	1	
4	Parallel Linkage	1	17

5	M3X6 Screw	1	
6	Right Wrist Joint	1	31
7	Wrist Joint Connector	1	18

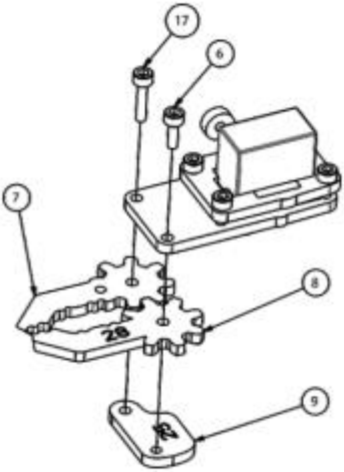
- After the first step, the gripper is assembled according to the pictures:

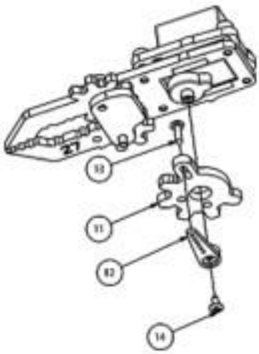
ATTACH THE GRIPPER SERVO



1.

- Tip: Never use super glue to assemble the chassis as it can make the acrylic more fragile and prone to damage.

2. 

3. 

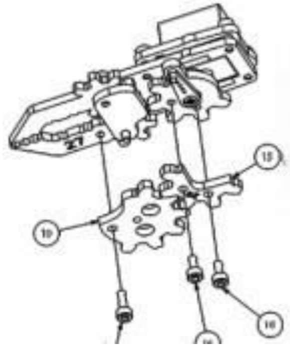
4. 

Figure 3.1.6.3, 3.1.6.4, 3.1.6.5, 3.1.6.6.: Steps to assemble the gripper of the arm respectively.

Item	Part Name	Quantity	Part No.
1	Right Wrist Attachment	1	23
2	Clamp Bottom Servo Mount	1	24
3	Left Wrist Attachment	1	22
4	SG90 Servo	1	
5	Clamp Top Servo Collar	1	21
6	M3X8 Screw	6	
7	Right Gripper	1	27
8	Left Gripper	1	28
9	Gripper Plate	1	25
10	Gripper Gear	1	26

11	Top Servo Gear	1	30
12	Servo Single Arm	1	
13	Servo Mount Screw	1	
14	Servo Screw	1	
15	Bottom Servo Gear	1	29
16	M3X6 Screw	2	
17	M3X12 Screw	1	

- After assembling the gripper, it should look similar to the below picture:

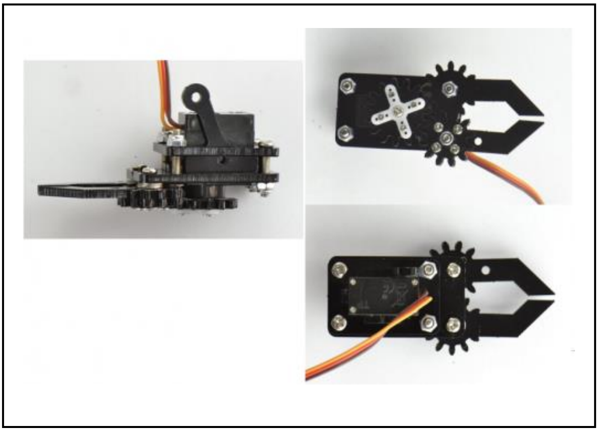
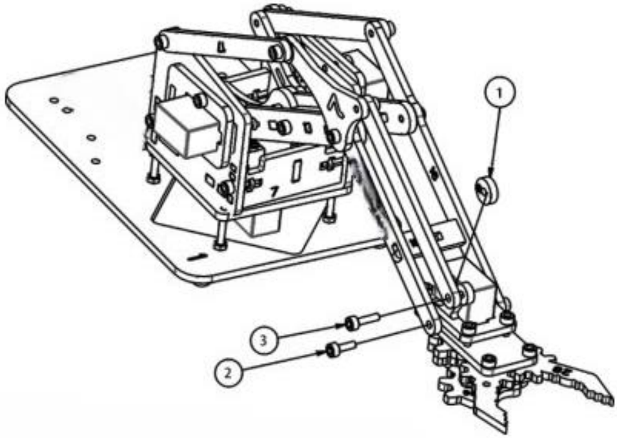


Figure 3.1.6.7.: The gripper after assembly. (This isn't the Endeavour's chassis, but a similar one.)

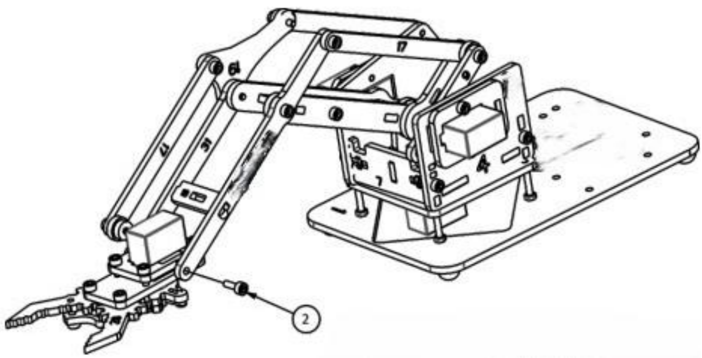
STEP 7: FINALIZATION

- This is the final step of assembling the robot arm and the easiest step of the assembly.
- Follow the pictures for the assembly:

ASSEMBLING ALL SEPARATE PARTS



1.



2.

Figure 3.1.7.1., 3.1.7.2.: Steps to finalize the arm respectively.

Item	Part Name	Quantity	Part No.
1	Spacer	1	20
2	M3X8 Screw	3	

- Now, you have completely assembled the arm and the arm should be like this picture:

Figure 3.1.7.3.: The Endeavour C-001 after assembly without electronic components.

TESTING AND CALIBRATION

STEP 1: UPLOAD THE CODE

```
#include <Servo.h>

Servo baseServo, shoulderServo, elbowServo, wristServo;

int basePos = 90;
int shoulderPos = 90;
int elbowPos = 90;
int wristPos = 90;

const int BASE_MIN = 0, BASE_MAX = 180;
const int SHOULDER_MIN = 0, SHOULDER_MAX = 180;
const int ELBOW_MIN = 0, ELBOW_MAX = 180;
const int WRIST_MIN = 0, WRIST_MAX = 180;

int targetBasePos, targetShoulderPos, targetElbowPos, targetWristPos;

bool isMovingBase = false, isMovingShoulder = false, isMovingElbow = false, isMovingWrist = false;

void moveServoSmoothly(Servo &servo, int ¤tPos, int targetPos, int step = 1) {
  targetPos = constrain(targetPos, 0, 180);
  if (currentPos != targetPos) {
    if (abs(currentPos - targetPos) <= step) {
      currentPos = targetPos; // Snap to target
    } else {
```

```
      currentPos += (currentPos < targetPos) ? step : -step;
    }
    servo.write(currentPos);
  }
}

void moveBase(int delta) {
  targetBasePos = constrain(basePos + delta, BASE_MIN, BASE_MAX);
  isMovingBase = true;
}

void moveShoulder(int delta) {
  targetShoulderPos = constrain(shoulderPos + delta, SHOULDER_MIN, SHOULDER_MAX);
  isMovingShoulder = true;
}

void moveElbow(int delta) {
  targetElbowPos = constrain(elbowPos + delta, ELBOW_MIN, ELBOW_MAX);
  isMovingElbow = true;
}

void moveWrist(int delta) {
  targetWristPos = constrain(wristPos + delta, WRIST_MIN, WRIST_MAX);
  isMovingWrist = true;
}

void setup() {
  baseServo.attach(8);
  shoulderServo.attach(9);
  elbowServo.attach(10);
  wristServo.attach(11);
}

void loop() {
```

```
    moveServoSmoothly(baseServo, basePos, 90);
    moveServoSmoothly(shoulderServo, shoulderPos, 90);
    moveServoSmoothly(elbowServo, elbowPos, 90);
    moveServoSmoothly(wristServo, wristPos, 90);
}

```

STEP 2: CONNECT THE SERVOS

- Connect the Servos according to the below table and turn on the Arduino board from a power source. All the Servos will go to their idle positions.
- Tip: Don't use the computer's USB port to power the Arduino to function properly as the computer's USB port can't provide enough current.

Servo Name	Connected Pin no.	Idle Position (in degrees)	Minimum Position	Maximum Position
Base	8	90	0	180
Shoulder	9	90	0	180
Elbow	10	90	0	180
Gripper	11	90	0	180

CHAPTER 4: PROGRAMMING THE 4DOF ROBOT ARM

ARM'S ELECTRONIC CONNECTIONS

CONNECTING THE ELECTRONIC COMPONENTS

- The circuitry is shown in this schematic and you have to connect the components according to the schematic.

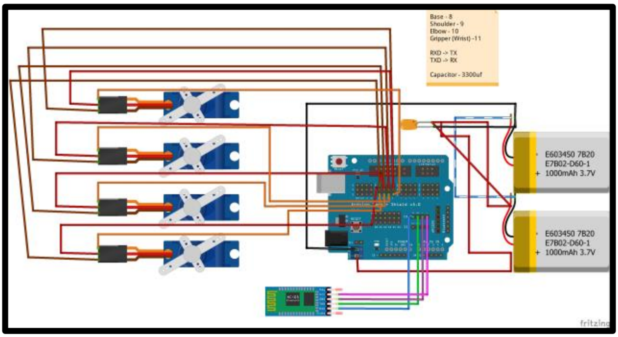


Figure 3.1.7.3.: The Endeavour - 4DOF Robot Arm's circuitry in Fritzing.

- The table below shows the pinout of the Servos and the position properties.

Servo Name	Connected Pin no.	Idle Position (in degrees)	Minimum Position	Maximum Position
Base	8	90	0	180
Shoulder	9	90	0	180
Elbow	10	90	0	180
Gripper	11	90	0	180

- This table shows the connections of the HC-05 Bluetooth module to the Arduino Uno.

Arduino Uno (Sensor Shield) - COM	HC-05 Bluetooth Module
TX	RXD
RX	TXD
-	GND
+	VCC

- Tip: Disconnect the HC-05 Bluetooth Module when uploading the code to the Arduino Uno to prevent malfunctions.

Figure 4.1.2: The QR Code to download the code from GitHub. Scan the code to get the code.

PROGRAMMING THE ARM

STEP 1: DOWNLOAD THE CODE

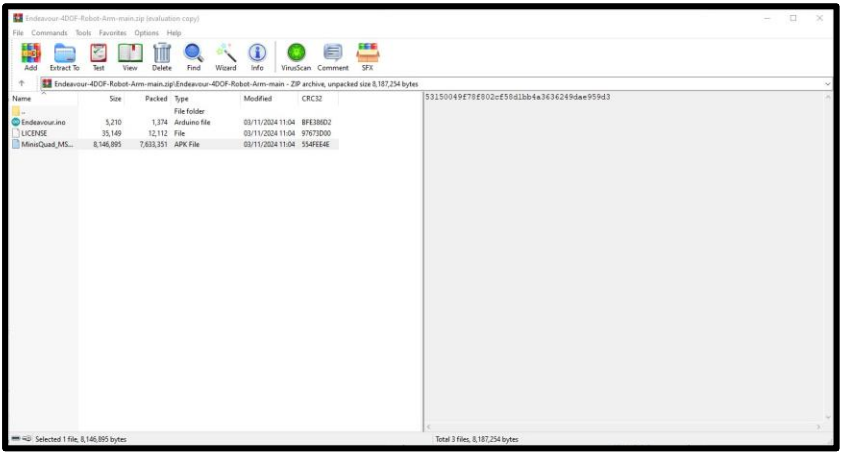


Figure 4.2.1.: The ".zip" file opened in WinRAR 6.1.0.

- Scan the QR Code to download the code and extract the ".zip" file that it comes in compressed.

STEP 2: UPLOAD THE CODE TO THE ARDUINO

- After extracting the ".zip" file, open the ".ino" file in Arduino IDE (if you haven't downloaded the Arduino IDE, download it to upload the code to the Arduino) and compile to test.

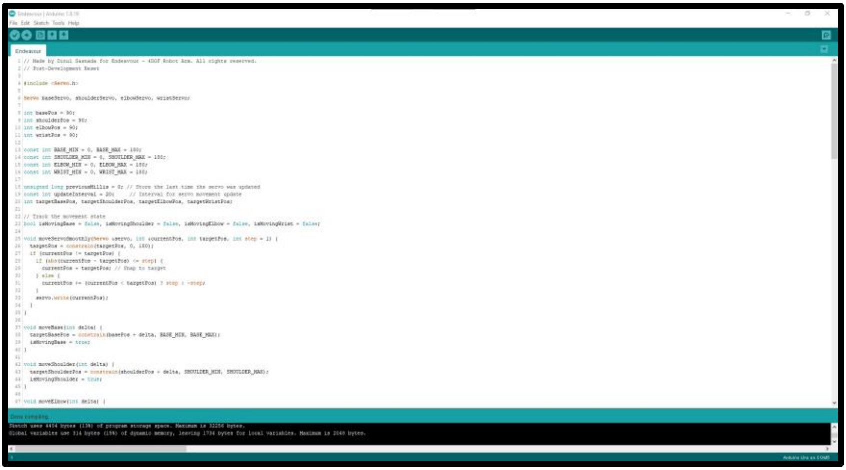


Figure 4.2.2.: The code has completed compiling successfully.

- If you bought an Arduino Uno SMD (Clone), then you may have to install additional drivers (CH340G USB

Serial-to-TTL Driver) in order to get the board to function.

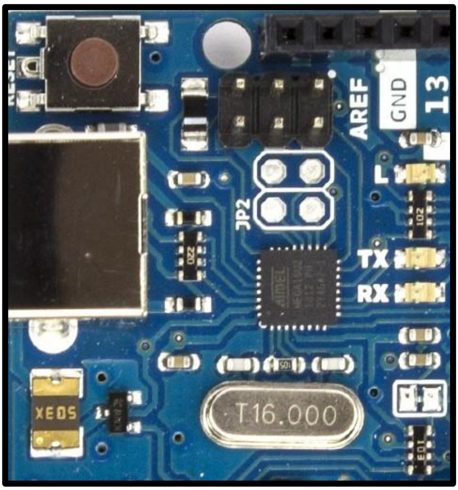
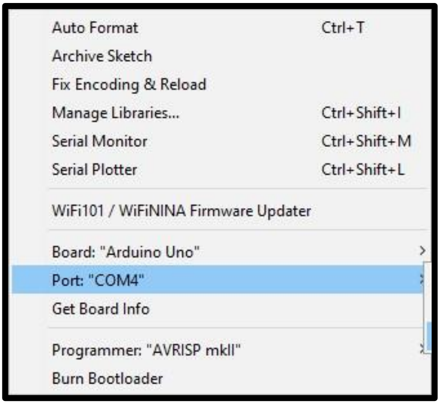


Figure 4.2.3, 4.2.4.: The Arduino Uno Clone (Top) with a WCH CH340G USB-to-TTL Converter and a genuine Arduino Uno (Bottom) with an ATMega 16U2 USB-to-TTL respectively.

- Connect your Arduino Uno to the computer with Arduino IDE using a USB B Male – USB A Female cable and select the COM port in the Arduino IDE.



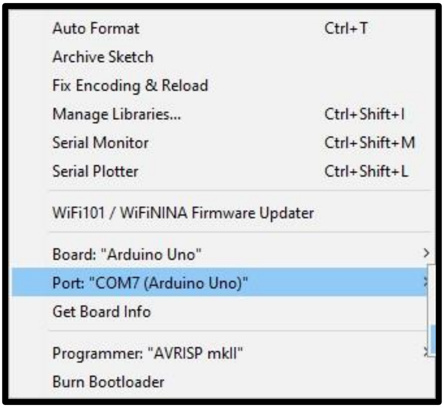


Figure 4.2.5, 4.2.6.: The Arduino Uno Clone (Top) with a WCH CH340G USB-to-TTL Converter and a genuine Arduino Uno (Bottom) with an ATMega 16U2 USB-to-TTL at the COM port selection screen respectively.

- Then click the Upload button at the top of the Arduino IDE window to upload the code to the Arduino Uno.

Figure 4.2.7.: The code has uploaded successfully to the Arduino.

- Don't forget to disconnect the HC-05 Bluetooth Module when uploading the code to the Arduino Uno to prevent malfunctions and to ensure a successful code uploading.
- Tip: Sometimes, you may need to disconnect the Servos from the Arduino Sensor Shield to properly upload the code if your computer's USB port can't provide enough power for the Servos to work. (This only applies to USB 2.0 ports as they only provide 500mA over USB 3.0's 900mA -1A current output.) So, it is recommended to connect the Arduino to a USB 3.0 port.

- After uploading, you can control the arm using the "Endeavour – 4DOF Robot Arm Controller" which will be discussed on the 5th Chapter.

TROUBLESHOOTING STEPS

- This section will help you with the common problems you'll have when uploading the code.

- The board doesn't show up on the COM port selection window: If it's an Arduino Uno Clone, install CH340G USB-to-TTL driver to the computer. If it is already installed, connect the board to a different USB port.
- The code doesn't upload: Disconnect the HC-05 Bluetooth module. If already disconnected, disconnect the Servos from the Arduino. Or else, connect the Arduino to a different USB port.
- The COM port not found: Reselect the COM port If it doesn't work, connect the board to a different USB port.
- Exit Status 1: Reselect the COM port or try re-uploading the code. Or, connect the Arduino to a different USB port and disconnect any connections to the board.

SPECIAL CODE FEATURES

1. Servo Control Setup:

- The code initializes servos for the base, shoulder, elbow, and wrist.
- Each servo has position limits (‘BASE_MIN’, ‘BASE_MAX’, etc.), with specific ranges for safety.

```
#include <Servo.h>

Servo baseServo, shoulderServo, elbowServo, wristServo;

int basePos = 90;
int shoulderPos = 90;
int elbowPos = 90;
int wristPos = 90;

const int BASE_MIN = 0, BASE_MAX = 180;
const int SHOULDER_MIN = 0, SHOULDER_MAX = 180;
const int ELBOW_MIN = 0, ELBOW_MAX = 180;
const int WRIST_MIN = 0, WRIST_MAX = 180;

```

2. Smooth Movement Implementation:

- A ‘moveServoSmoothly’ function ensures each servo moves smoothly to its target position by adjusting the position gradually, avoiding abrupt jumps.

- This function also includes a "snap" feature that aligns the servo directly with the target when it's close, maintaining precision.

```
void moveServoSmoothly(Servo &servo, int ¤tPos, int targetPos, int step = 1) {
  targetPos = constrain(targetPos, 0, 180);
  if (currentPos != targetPos) {
    if (abs(currentPos - targetPos) <= step) {
      currentPos = targetPos; // Snap to target
    } else {
      currentPos += (currentPos < targetPos) ? step : -step;
    }
    servo.write(currentPos);
  }
}

```

3. Individual Movement Functions:

- Separate functions (e.g., ‘moveBase’, ‘moveShoulder’) constrain each joint’s movement within set ranges, enhancing safety and preventing over-extension.

```
void moveBase(int delta) {
  targetBasePos = constrain(basePos + delta, BASE_MIN, BASE_MAX);
  isMovingBase = true;
}

void moveShoulder(int delta) {
  targetShoulderPos = constrain(shoulderPos + delta, SHOULDER_MIN, SHOULDER_MAX);
  isMovingShoulder = true;
}

```

```
void moveElbow(int delta) {
  targetElbowPos = constrain(elbowPos + delta, ELBOW_MIN, ELBOW_MAX);
  isMovingElbow = true;
}

void moveWrist(int delta) {
  targetWristPos = constrain(wristPos + delta, WRIST_MIN, WRIST_MAX);
  isMovingWrist = true;
}

```

4. Timing Control:

- A timing variable (‘previousMillis’) and interval (‘updateInterval’) manage the update rate, ensuring consistent movement without overwhelming the servos.

```
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= updateInterval) {
    previousMillis = currentMillis; // Update the last update time

    // Move servos smoothly
    if (isMovingBase) {
      moveServoSmoothly(baseServo, basePos, targetBasePos);
      if (basePos == targetBasePos) isMovingBase = false;
    }
    if (isMovingShoulder) {
      moveServoSmoothly(shoulderServo, shoulderPos, targetShoulderPos);
      if (shoulderPos == targetShoulderPos) isMovingShoulder = false;
    }
```

```
    if (isMovingElbow) {
      moveServoSmoothly(elbowServo, elbowPos, targetElbowPos);
      if (elbowPos == targetElbowPos) isMovingElbow = false;
    }
    if (isMovingWrist) {
      moveServoSmoothly(wristServo, wristPos, targetWristPos);
      if (wristPos == targetWristPos) isMovingWrist = false;
    }
  }

```

THE ENTIRE CODE

```
// Made by Dinul Sasnada for Endeavour - 4DOF Robot Arm. All rights reserved.
// Post-Development Reset

#include <Servo.h>

Servo baseServo, shoulderServo, elbowServo, wristServo;

int basePos = 90;
int shoulderPos = 90;
int elbowPos = 90;
int wristPos = 90;

const int BASE_MIN = 0, BASE_MAX = 180;
const int SHOULDER_MIN = 0, SHOULDER_MAX = 180;
const int ELBOW_MIN = 0, ELBOW_MAX = 180;
```

```
const int WRIST_MIN = 0, WRIST_MAX = 180;

unsigned long previousMillis = 0; // Store the last time the servo was updated
const int updateInterval = 20;     // Interval for servo movement update
int targetBasePos, targetShoulderPos, targetElbowPos, targetWristPos;

// Track the movement state
bool isMovingBase = false, isMovingShoulder = false, isMovingElbow = false, isMovingWrist = false;

void moveServoSmoothly(Servo &servo, int ¤tPos, int targetPos, int step = 1) {
  targetPos = constrain(targetPos, 0, 180);
  if (currentPos != targetPos) {
    if (abs(currentPos - targetPos) <= step) {
      currentPos = targetPos; // Snap to target
    } else {
      currentPos += (currentPos < targetPos) ? step : -step;
    }
    servo.write(currentPos);
  }
}

void moveBase(int delta) {
  targetBasePos = constrain(basePos + delta, BASE_MIN, BASE_MAX);
  isMovingBase = true;
}

void moveShoulder(int delta) {
  targetShoulderPos = constrain(shoulderPos + delta, SHOULDER_MIN, SHOULDER_MAX);
  isMovingShoulder = true;
}

```

```
void moveElbow(int delta) {
  targetElbowPos = constrain(elbowPos + delta, ELBOW_MIN, ELBOW_MAX);
  isMovingElbow = true;
}

void moveWrist(int delta) {
  targetWristPos = constrain(wristPos + delta, WRIST_MIN, WRIST_MAX);
  isMovingWrist = true;
}

void P() {
  Serial.write(wristPos);
  Serial.write(elbowPos);
  Serial.write(shoulderPos);
  Serial.write(basePos);
}

void R() {
  // Write to servos directly to reset
  moveBase (90 - basePos);
  moveShoulder(90 - shoulderPos);
  moveElbow(90 - elbowPos);
  moveWrist(90 - wristPos);
}

void K() {
  baseServo.detach();
  shoulderServo.detach();
  elbowServo.detach();
  wristServo.detach();

}

void Q() {
  baseServo.attach(8);
  shoulderServo.attach(9);
  elbowServo.attach(10);
```

```
  wristServo.attach(11);
}

void setup() {
  Serial.begin(9600);

  baseServo.attach(8);
  shoulderServo.attach(9);
  elbowServo.attach(10);
  wristServo.attach(11);

  // Move servos smoothly to their initial positions
  moveServoSmoothly(baseServo, basePos, 90);
  moveServoSmoothly(shoulderServo, shoulderPos, 90);
  moveServoSmoothly(elbowServo, elbowPos, 90);
  moveServoSmoothly(wristServo, wristPos, 90);

  Serial.println("Endeavour - 4DOF Robot Arm Ready!");
}

void loop() {
  if (Serial.available() > 0) {
    char value = Serial.read();
    switch (value) {
      case 'A': moveBase(-10); break;
      case 'B': moveBase(10); break;
      case 'C': moveShoulder(-5); break;
      case 'D': moveShoulder(5); break;
      case 'E': moveElbow(-5); break;
      case 'F': moveElbow(5); break;
      case 'G': moveWrist(-5); break;
      case 'H': moveWrist(5); break;
      case 'K': K(); break;
      case 'M': moveWrist(180 - wristPos); break;
      case 'P': P(); break;
      case 'Q': Q(); break;
      case 'R': R(); break;
      case 'S': moveBase(180 - basePos); break;
```

```
        case 'T': moveShoulder(180 - shoulderPos); break;
        case 'U': moveElbow(180 - elbowPos); break;
        case 'V': moveBase(basePos = 0); break;
        case 'X': moveShoulder(shoulderPos = 60); break;
        case 'Y': moveElbow(elbowPos = 0); break;
        case 'Z': moveWrist(wristPos = 0); break;
        case 'a': baseServo.detach(); break;
        case 'b': shoulderServo.detach(); break;
        case 'c': elbowServo.detach(); break;
        case 'd': wristServo.detach(); break;
        case 'e': baseServo.attach(8); break;
        case 'f': shoulderServo.attach(9); break;
        case 'g': elbowServo.attach(10); break;
        case 'h': wristServo.attach(11); break;
        case 'i': moveBase(90 - basePos); break;
        case 'j': moveShoulder(90 - shoulderPos); break;
        case 'k': moveElbow(90 - elbowPos); break;
        case 'l': moveWrist(90 - wristPos); break;
        default: Serial.println("Invalid Command..."); break;
      }
    }

    unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= updateInterval) {
      previousMillis = currentMillis; // Update the last update time

      // Move servos smoothly
      if (isMovingBase) {
        moveServoSmoothly(baseServo, basePos, targetBasePos);
        if (basePos == targetBasePos) isMovingBase = false;
      }
```

```
    if (isMovingShoulder) {
      moveServoSmoothly(shoulderServo, shoulderPos, targetShoulderPos);
      if (shoulderPos == targetShoulderPos) isMovingShoulder = false;
    }
    if (isMovingElbow) {
      moveServoSmoothly(elbowServo, elbowPos, targetElbowPos);
      if (elbowPos == targetElbowPos) isMovingElbow = false;
    }
    if (isMovingWrist) {
      moveServoSmoothly(wristServo, wristPos, targetWristPos);
      if (wristPos == targetWristPos) isMovingWrist = false;
    }
  }
}

```

CHAPTER 5: DEVELOPING AN ANDROID APP FOR CONTROL

TESTING AND DEBUGGING THE APP

- The "Endeavour – 4DOF Robot Arm Controller" is built using the Android Studio with Jave and Kotlin programming languages for devices with Android 44 (API Level 29) or higher.
- The app is tested in real-life devices such as Samsung T235Y, Redmi A1+, Redmi Note 8, and Oppo A38, on Android Virtual Devices (AVDs) such as Google Pixel 3a, Google Nexus 7, and in a VMWare Virtual Machine running Android 9.0 to ensure proper functioning on different devices with different screen sizes, specifications, and Android version, ensuring compatibility with a broad range of devices.
- The app is ensured to have no loopholes or flaws in its code for reliable and proper functioning.
- The app is also tested with its position reading accuracy according to the button clicks.
- The "Endeavour – 4DOF Robot Arm Controller" is a released app, not just a debug version, and has all the features implemented and debugged for reliability.

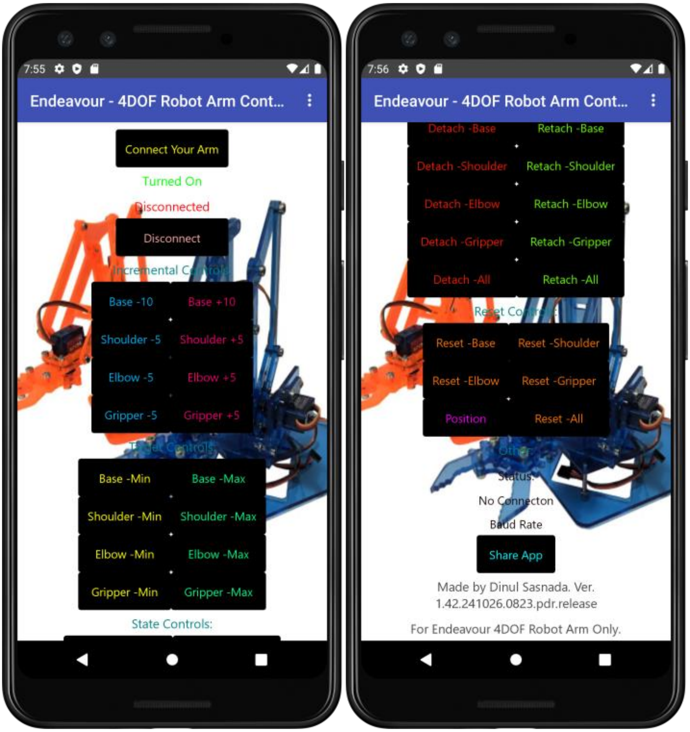


Figure 4.1.1, 4.1.2.: The "Endeavour – 4DOF Robot Arm Controller" running on a Google Pixel 3a (API Level 30) emulated in Android Studio, startup screen (left), bottom (right).

USING THE APP

- Install the "Endeavour – 4DOF Robot Arm Controller" from the below QR Code and install the app. You may need to be permitted to install the app because

Android doesn't like apps from outer sources that aren't verified by Play Protect.

- There's no iOS version of this app and it's unclear to say that there will be one as it requires some serious modifications to the app.

Figure 4.2.1.: The QR Code to download the "Endeavour – 4DOF Robot Arm Controller" app.

- Tip: Always turn on Bluetooth before opening the "Endeavour – 4DOF Robot Arm Controller" app so that you can connect without hassle. If you turn on Bluetooth after opening the app, you may need to restart the app to connect to a device

STEP 1: PAIR THE BLUETOOTH MODULE WITH THE PHONE

- Turn on the Endeavour arm and check whether the HC-05's LED Blinks continuously.
- If it blinks, turn on Bluetooth on your smartphone and scan for devices. It may take some time as the HC-05 uses a very old Bluetooth protocol (v 2.0+EDR). Sometimes, it is in the *"Rarely Used Devices"* section.
- When the HC-05 shows up, tap on it to connect and it will ask for a passkey, which is almost always "1234" and pair with the HC-05.

STEP 2: CONNECT TO THE ARM

- After pairing, open the "Endeavour – 4DOF Robot Arm Controller" app click on *"Connect Your Arm"* and select the "[MAC Address] HC-05" from the list to connect.
- Then, you'll see that the label that says *"Disconnected"* now says *"Connected"* and you can see the MAC Address and the device name on the bottom of the screen.
- Now you can control the arm from the app and when you need to disconnect, just click the *"Disconnect"* button.

Figure 5.2.1.1.: The paired Bluetooth device list (the MAC Addresses are erased for privacy reasons).

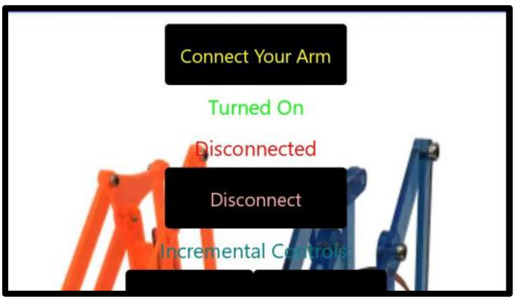


Figure 5.2.1.2.: The "Disconnect" label and the Bluetooth Status label.

FUNCTIONS OF BUTTON CLUSTERS

CONNECTION CONTROLS

- Used for things related to Bluetooth Connection.
- Consists of "Connect" and "Disconnect" buttons.

INCREMENTAL CONTROLS

- Used to change the degrees the arm's Servos have moved according to the below table.

Servo Name	Increments (in degrees)
Base	+10 and -10
Shoulder	+5 and -5
Elbow	+5 and -5
Gripper	+5 and -5

TARGET CONTROLS

- Used to move the Servos to their minimum possible and maximum possible positions respectively. The positions are in the below table

Servo Name	Minimum Position (In degrees)	Maximum Position (In degrees)
Base	0	180
Shoulder	0	180
Elbow	0	180
Gripper	0	180

STATE CONTROLS

- This is the kill (detach) switch of each Servos and you can also re-attach these Servos from the Arduino from this cluster of buttons.
- You can either detach or re-attach all the Servos at once or you can do it one-by-one.

RESET CONTROLS

- Used to reset all the Servos at once or one by one and to read the positions of the Servos. The reset positions are on this table:

Servo Name	Idle Position (in degrees)
Base	90
Shoulder	90
Elbow	90
Gripper	90

TROUBLESHOOTING STEPS

- This section will provide the solutions for common problems you'll have with the app.

- No Bluetooth devices show up on the list: Turn on Bluetooth and restart the app. If it persists, restart the device.

- The device is not shown: Make sure that the device is paired with the device before. If not, pair it with the Bluetooth module and restart the app.

- The device doesn't connect to the arm: Make sure the arm is turned on and the Bluetooth module's LED blinks If not, hard-restart the arm.

- The arm doesn't respond to commands: Make sure the arm is connected and if not, restart the arm and the app.

CHAPTER 6: PRACTICAL APPLICATIONS AND FUTURE DIRECTIONS

REAL-WORLD APPLICATIONS OF ROBOTIC ARMS

- This part of the chapter covers how a 4-degree-of-freedom robot arm can be applied in various fields.
  Education and Learning: A 4DOF robot arm is a great tool for education, especially in STEM (Science, Technology, Engineering, and Mathematics) fields. It allows students and hobbyists to get hands-on experience with:
  Mechanics and Electronics: Assembling and programming the robot provides insight into mechanical design, electronic circuitry, and control systems.
  Programming and Coding: Controlling the arm through code (for example, in Arduino or Python) teaches basic programming, logical thinking, and problem-solving.
  This makes it an accessible introduction to robotics for students, hobbyists, and even early researchers, helping them bridge the gap between theory and practice.

Industrial and Manufacturing Applications: In industrial settings, robot arms with several degrees of freedom are used for:

Pick-and-Place Operations: These are repetitive tasks where the robot arm picks up objects and places them in a different location. For example, assembling small parts on a manufacturing line or sorting products by category.

Assembly Line Tasks: A 4DOF arm can be programmed to handle small components, apply glue, screw parts together, or perform other simple tasks that require repeated precise movements.

Though small and lightweight, your robot arm models the same basic principles used in industry, helping readers understand how robots improve efficiency and consistency in production lines.

Assistance in Research and Prototyping: Robot arms are useful tools for prototyping and research. For example:

Lab Automation: Robotic arms can handle repetitive lab tasks, like mixing liquids or positioning test samples, freeing researchers from repetitive, manual work.

Prototyping Larger Systems: Smaller robot arms like Endeavour allow engineers and designers to test movement patterns, design ideas, and interactions before investing in larger, more expensive machines.

Endeavour can demonstrate these concepts on a small scale, showing how robot arms are valuable assets in experimental setups and early-stage product development.

DIY Projects and Hobbyist Uses: DIY and hobbyist communities use robotic arms in creative projects, such as:
Home Automation: People might use robot arms to automate tasks around the house, like sorting objects, pressing buttons, or even assisting in the kitchen.
Artistic Endeavors: Robotics enthusiasts can program arms for precise actions like drawing, painting, or creating 3D-printed objects.

PROJECT EXTENSIONS AND THE FUTURE

- This section focuses on the potential improvements and advancements that can make Endeavour more versatile, effective, and functional.

- Improving Structural Components
  Upgrading materials and precision parts can make the arm stronger, more accurate, and more durable:
  Stronger Materials: Replacing acrylic with aluminum or carbon fiber could improve the arm's strength, allowing it to lift heavier loads without structural risk.
  Precision Parts: Higher-quality servos or stepper motors could increase the accuracy of movements, making the arm more suitable for delicate tasks.

These improvements make the arm more practical for a wider range of tasks, from educational demonstrations to light-duty industrial use.

- Battery and Power Management Enhancements
  Power management is essential for long-term operation. Suggestions include:
  Larger Batteries: Higher-capacity batteries would extend the operational time, which is essential for tasks requiring continuous activity.
  Efficient Power Circuits: By optimizing the power circuit, the robot can operate for longer without sacrificing performance, which is ideal for autonomous or remote tasks and better power management enables the arm to function more independently, reducing the need for frequent recharging or supervision.

- Adding More Degrees of Freedom
  Adding more joints to the robot arm increases its range of motion, making it capable of reaching more complex angles and positions. With each added degree of freedom:
  Greater Flexibility: The arm can reach around obstacles or approach objects from various directions, similar to how a human arm has multiple joints.
  Expanded Applications: With more freedom of movement, the arm can handle tasks that require intricate positioning, such as in welding or assembly.

CHAPTER 7: CONCLUSION AND REFLECTIONS

CONCLUSION

- In developing Endeavour, we've explored both the challenges and rewards of building a functional, adaptable robot arm. This project demonstrates how robotics can bridge the gap between theoretical knowledge and hands-on skills, bringing engineering concepts to life. Endeavour's design, while simple in structure, has significant potential for practical applications, from basic tasks to more complex interactions when integrated with sensors and intelligent control systems. Looking forward, there are endless possibilities for expanding Endeavour's capabilities, like adding automation features or even exploring AI-based movement control. This journey with Endeavour has laid a strong foundation for future projects, inspiring continuous learning and innovation in robotics.

SOME MEMORIES IN THIS JOURNEY

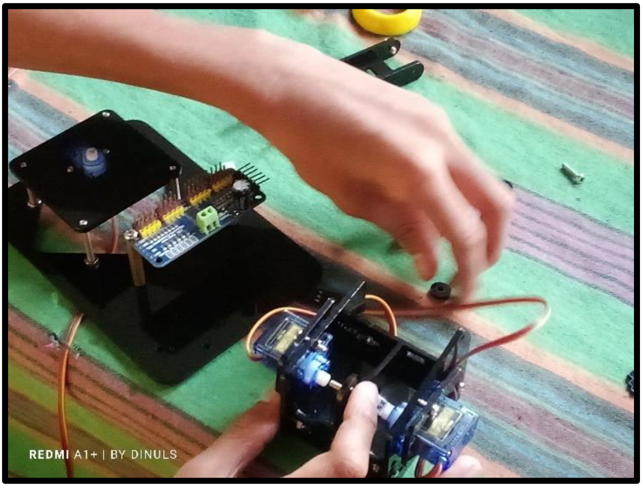


Figure 7.2.1, 7.2.2.: The remnants of the Endeavour's chassis C-001 after lifting 90 grams and a lot of super glue (top), assembling the C-001 (bottom) respectively.

8. GLOSSARY AND APPENDICES

- https://www.scribd.com/document/579944785/Robotic-Arm-4DOF/
- https://github.com/MeArm/MeArm/
- Grade 9 ICT Textbook (2018), Education Publication Department
- https://www.sgbotic.com/index.php?dispatch=products.view&product_id=1939/
- https://www.mokosmart.com/guide-on-different-bluetooth-versions/
- https://wiki.keyestudio.com/Ks0198_keyestudio_4DOF_Robot_Mechanical_Arm_Kit_for_Arduino_DIY/
- https://github.com/dinul1/Endeavour-4DOF-Robot-Arm/