BMS COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

A Technical Seminar Report based on Technical Activity

DIGITAL TWIN OF A ROBOTIC ARM

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering in
Computer Science and Engineering

Submitted by:
SAFWAN MOHAMMED HANIF 1BM21CS180

Work carried out at

Internal Guide

Dr. Nagarathna N
Assistant Professor
BMS College of Engineering

Department of Computer Science and Engineering
BMS College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
2022-2023

BMS COLLEGE OF ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I, SAFWAN MOHAMMED HANIF (1BM21CS180) student of 4thSemester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this technical seminar entitled "DIGITAL TWIN" has been carried out under the guidance Dr. Nagarathna N, Professor, Department of CSE, BMS College of Engineering, Bangalore during the academic semester March-July 2023. I also declare that to the best of my knowledge and belief, the technical seminar report is not from part of any other report by any other students.

Signature of the Candidate

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Technical Seminar titled "DIGITAL TWIN**"** has been carried out by SAFWAN MOHAMMED HANIF (1BM21CS180) during the academic year 2022-2023.

Signature of the Guide

Signature of the Head of the Department

Signature of Examiners with date

1. Internal Examiner _____
2. External Examiner _____

Abstract

Digital twins are an emerging technology that has gained a lot of attention from industry and in recent years academia. Industry 4.0 ideas have made it easier for digital twins to grow, especially in the manufacturing sector. Digital twins can be defined in many ways but can best be described as the seamless integration of data in either direction between a physical machine and a virtual machine.This paper discusses the challenges, applications and enabling technologies of Artificial Intelligence, IoT and Digital twins. It reviews publications related to Digital twins and provides a categorical overview of recent papers on the topic. The paper has categorised papers on the topic by research areas (manufacturing, healthcare and smart cities) and discusses a number of papers on the topic that reflect the state of research on the topic. This paper provides an overview of enabling technologies and challenges for Digital twins, as well as open research on the subject.

.

Table of Contents

Chapter 1: INTRODUCTION

1.1 OVERVIEW

Digital Twins, a concept rapidly gaining prominence, are virtual replicas of physical entities or systems that incorporate real-time data, advanced modeling, analytics, and simulation capabilities. They offer a transformative approach to understanding, monitoring, and optimizing a wide range of physical phenomena, spanning industries from manufacturing and healthcare to smart cities and agriculture. In this report, we explore the fundamental components of Digital Twins, including their physical counterparts, digital representations, data integration, analytics, and communication infrastructure. We delve into the various types of Digital Twins, highlighting their diverse applications, such as improving manufacturing efficiency, enabling personalized medical treatments, optimizing urban infrastructure, and enhancing resource management. Additionally, we discuss the manifold benefits Digital Twins bring, including data-driven decision-making, predictive maintenance, accelerated innovation, and improved resource allocation. In a world increasingly driven by data and simulation, understanding Digital Twins is essential for businesses and industries seeking to harness their transformative potential for enhanced efficiency and innovation.

1.2 MOTIVATION

Digital Twins lies in its potential to revolutionize how we understand, interact with, and optimize the physical world in an increasingly digital age. With the advent of advanced technologies like the Internet of Things (IoT), big data analytics, and high-performance computing, Digital Twins have emerged as a powerful bridge between the physical and digital realms. Businesses and industries are recognizing the need to harness the capabilities of Digital Twins to gain a competitive edge, improve operational efficiency, reduce costs, and enhance decision-making processes. Whether it's streamlining manufacturing operations, transforming healthcare through personalized treatments, or making cities smarter and more sustainable, Digital Twins offer a transformative approach that can significantly impact our daily lives. As we delve into the components, types, applications, and benefits of Digital Twins, this report aims to provide a compelling motivation for embracing this concept as a pivotal force shaping the future of various industries and technologies.

1.3 OBJECTIVE

The objectives of this report are multifaceted, encompassing both an in-depth understanding of Digital Twins and their practical implications across industries.

Firstly, we aim to provide a comprehensive educational resource that elucidates the core components of Digital Twins, ranging from the physical entities they replicate to the sophisticated digital modeling techniques used. By offering a clear and concise explanation of these components, our objective is to

demystify the concept of Digital Twins, making it accessible to a wide audience, from technology enthusiasts to industry professionals.

Secondly, we intend to shed light on the various types of Digital Twins, showcasing their versatility and adaptability across different domains. Whether it's product design and testing, healthcare simulations, or urban planning, we aim to illustrate how Digital Twins can be tailored to meet specific industry needs. Our objective here is to inspire readers with a diverse array of use cases, sparking innovative thinking and encouraging the exploration of novel applications.

Additionally, our report seeks to emphasize the practical benefits that Digital Twins offer. We aim to elucidate how they can facilitate data-driven decision-making through real-time monitoring and predictive analytics. Furthermore, we will explore how Digital Twins can lead to cost savings by enabling predictive maintenance and improving resource allocation. Our objective is to highlight the tangible advantages that organizations can gain by embracing this technology, promoting its adoption as a strategic tool for achieving operational excellence.

Lastly, we aim to address the challenges and considerations associated with implementing Digital Twins. By offering insights into potential pitfalls, security concerns, and scalability issues, our objective is to provide a balanced perspective that equips readers with the knowledge to navigate the complexities of integrating Digital Twins into their operations.

In summary, our report's objectives encompass education, inspiration, and practical insights, with the ultimate goal of fostering a deeper understanding of Digital Twins and their transformative potential in the contemporary technological landscape.

CHAPTER 2: LITERATURE SURVEY

The concept of Digital Twins originated from the manufacturing industry and has since been adopted across various sectors. Digital Twins are virtual representations of physical objects or systems, offering real-time insights and simulation capabilities. Understanding their foundational principles is crucial to grasp their significance. Digital Twins have found applications in manufacturing, healthcare, aerospace, automotive, energy, and smart cities. In manufacturing, they optimize production processes. In healthcare, they enable personalized treatments. In aerospace and automotive, they aid in design and maintenance. In energy, they improve resource management, and in smart cities, they enhance urban planning and infrastructure maintenance. Key technologies enabling Digital Twins include IoT devices, data analytics, simulation tools, and cloud computing. These technologies provide the data integration and computational power required to create and operate Digital Twins effectively. Implementing Digital Twins comes with challenges such as data privacy, security, scalability, and interoperability. Organizations must navigate these issues to ensure the successful deployment of Digital Twins. Emerging trends in the Digital Twins field include edge computing, explainable AI, and blockchain integration. These trends are shaping the future of Digital Twins by enhancing their capabilities and applications. Real-world case studies and practical implementations showcase how organizations are leveraging Digital Twins to achieve specific goals and benefits. These examples provide valuable insights into the practical aspects of Digital Twin adoption. Ethical considerations, particularly in healthcare and data privacy, are essential when implementing Digital Twins. Organizations must also comply with data protection regulations, such as GDPR, to ensure responsible and legal use of data in Digital Twins.

CHAPTER 3: METHODOLOGY/TECHNIQUES OR ALGORITHM USED

3.1 Procedure for Creating a Digital Twin

1) Identify the physical object or system that you want to create a digital twin of. This could be anything from a simple machine to a complex manufacturing process.

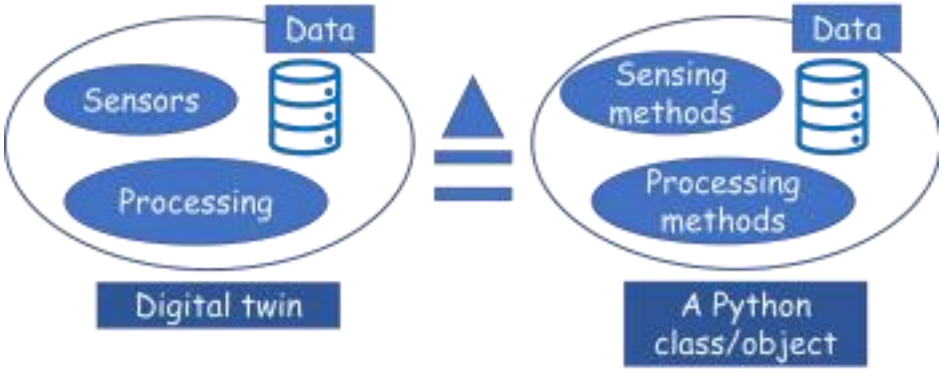
2) Collect data about the physical object or system. This data can come from a variety of sources, including sensors, cameras, and other data acquisition devices.

3) Use modeling software to create a virtual representation of the physical object or system. This model should be as accurate as possible and include all relevant details.

4) Integrate the data you collected in step 2 into the virtual model. This will make the model more accurate and allow it to accurately simulate the behavior of the physical object or system.

5) Test the digital twin. Use the digital twin to simulate various scenarios and analyze the results. This will help you identify potential problems, optimize performance, and develop new products and services.

6) Continuously update the digital twin. As new data becomes available or the physical object or system changes, update the digital twin to ensure it remains accurate and useful.



We can emulate the sensors and data processors with suitable methods/functions, store the gathered data in a database or internal variables, and encapsulate everything into a Python class.Once we accomplish that, we can also hope that the Python object

• can be used in suitable simulation programs,
• can be probed for data, and
• can even be subject to optimization routines for enhancing suitable internal parameters

For example, let's take an example of a digital switch. This example has been taken from a research paper whose reference is given below. The switch has two variables defining its function, Threshold voltage(Vth) and Breakdown Voltage (BV).

3.2 ALGORITHM

STEP 1:

We create a python class encapsulating all the variables and methods that define its functionality.

```
class MOSFET:
    def __init__(self,params=None,terminals=None):

        # Params
        if params is None:
            self._params_ = {'BV':28,
                             'Vth':1.0,
                             'gm':1e-2}
        else:
            self._params_ = params

        # Terminals
        if terminals is None:
            self._terminals_ = {'source':0.0,
                        'drain':0.0,
                        'gate':0.0}
        else:
            self._terminals_ = terminals
```

STEP 2:

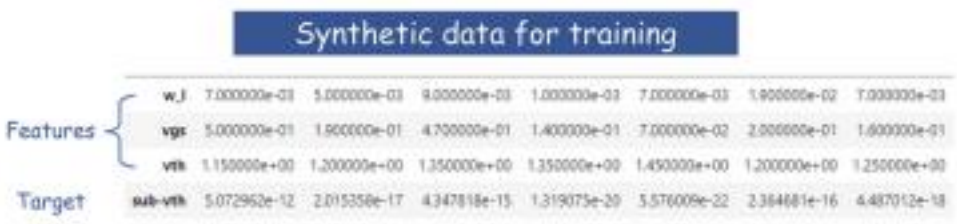
We define a method to calculate the Drain-to-Source current for the ON-state Switch using a simple, first-order analytical model. We will be checking if the Digital Twin can generate current-voltage characteristics similar to a physical device.

```
def id_vd(self,vgs=None,vds=None,rounding=True):
        """
        Calculates drain-source current from
        terminal voltages and gm
        """
        <code>
        if state=='ON':
            if vds <= vgs - vth:
                ids = self._params_['gm']*(vgs - vth - (vds/2))*vds
            else:
                ids = (self._params_['gm']/2)*(vgs-vth)**2

        <more code...>
```

STEP 3:

We apply suitable machine learning model to calculate the next output parameter which is 'Sub-threshold leakage'

The choice whether to use an analytical or ML model is subjective. However, the beauty of a digital twin is that it allows you to swap out an analytical model with an ML model (and vice versa) anytime you fancy. The choice of the model training is entirely flexible. For this demo, we created some synthetic data using a special helper function to train the model.



However, in real life, the semiconductor switch will run physical tests (using other machines) and record their leakage current for various Vgs bias voltages. It will also record the Vth of those MOSFETs in the test. All these data will flow into the Digital Twin and the model will be continuously trained and updated. After this model is trained, we can run various test cases to simulate and test variables and determine what leads to an optimum output and run various simulations

3.3 APPLICATIONS

1) Predicting the Performance of Packaging Materials

Food packaging is one of the primary offenders when it comes to single-use packaging, which has long been the misery of the environment. In order to tackle this issue , digital twins of the packaging material has been implemented in order to learn the consequences of the degradation of the single time use material on the environment and also to come up with a better alternative which is economic as well as eco friendly. Such kinds of digital twins are termed as Material Digital Twins. For Example , the well known Swedish Company , IKEA has implemented this kind of Digital Twin and has swapped their styrofoam packaging, which takes centuries to decompose, for a biodegradable mushroom-based material that breaks down at a much faster rate. Also , the material digital twins can track any damage to fleets of reusable containers and identify any design defects that need to be fixed for upcoming deliveries.

2) Improvement of Shipment Protection

Companies can learn how various packing circumstances may impact a certain product even before the first delivery is made by merging product and packaging data. By using the best packing, providers may reduce product damage. Sensors that transmit several data points throughout the course of an actual shipment collect the data for a digital twin shipment. The Digital Twin continue to add more value by collecting the data from the sensors , which helps to identify the weakness in the transportation process , allowing them to safeguard and boost future operations.

3) Supporting Logistics Infrastructure Via Design and Performance Optimization

The main advantage of creating a digital twin of the warehouse is that it helps in the continuous improvement of the warehouse over the long run. This helps the company to quickly find operational issues if any. It also helps in designing of various components inside the warehouse and also gives insights on effective usage of available space. By transferring the data into the model continuously it can also alert about any system failure or component failure with which many losses can be avoided , thus increasing the safety measures. Before making the real modifications, businesses may use digital twins to simulate the effects of layout changes, the addition of new machinery, and new processes

4) Integrate Operations Management At Major Global Hubs

The Operational Management at the big Harbor/Ports is often quite complicated , the arrival time , the departure time , sea water level , maximum cargo that can be loaded , etc many factors have to be taken into account. All these factors play an important role since proper transportation of the cargo has to be ensured, in order to avoid huge losses. This can be tackled by implementing a Digital Twin of the harbor. This can be used to simulate various scenarios and improve the operational efficiency. With the help of IoT technology , sensors can be used to detect weather conditions , wind speed , sea water level and many other factors , using which the arrival time , departure time of the ships and the quantity of the cargo to be loaded (depending on the weather conditions) can be planned out beforehand , and if any unexpected circumstances were to occur (which can be predicted by the ML model based on the continuous data that is being transferred) precautionary measures can be taken beforehand. A study in the US shows that when this project is implemented successfully they can save around $80,000 per hour!

CHAPTER 4: TOOLS USED

Unity, a versatile game development engine, plays a pivotal role in crafting Digital Twins. It empowers developers to create lifelike 3D simulations with real-time interactivity, physics modeling, and seamless integration of sensor data. Unity's cross-platform capabilities and collaborative tools enable the construction of immersive, interactive replicas of physical entities and systems, making it a valuable asset for Digital Twin development across various industries.

CHAPTER 5: MODULES IMPLEMENTED AND OUTPUT

5.1 C# CODES

CONTROLLER.CS

```
using System;
using Unity.Robotics;
using UnityEngine;

namespace Unity.Robotics.UrdfImporter.Control
{
    public enum RotationDirection { None = 0, Positive = 1, Negative = -1 };
    public enum ControlType { PositionControl };

    public class Controller : MonoBehaviour
    {
        private ArticulationBody[] articulationChain;
        private Color[] prevColor;
        private int previousIndex;

        [InspectorReadOnly(hideInEditMode: true)]
        public string selectedJoint;
        [HideInInspector]
        public int selectedIndex;

        public ControlType control = ControlType.PositionControl;
        public float stiffness;
        public float damping;
        public float forceLimit;
        public float speed = 5f; // Units: degree/s
        public float torque = 100f; // Units: Nm or N
        public float acceleration = 5f;// Units: m/s^2 / degree/s^2
```

```
public Color highLightColor = new Color(1.0f, 0, 0, 1.0f);

void Start()
{
    previousIndex = selectedIndex = 1;
    this.gameObject.AddComponent<FKRobot>();
    articulationChain = this.GetComponentsInChildren<ArticulationBody>();
    int defDyanmicVal = 10;
    foreach (ArticulationBody joint in articulationChain)
    {
        joint.gameObject.AddComponent<JointControl>();
        joint.jointFriction = defDyanmicVal;
        joint.angularDamping = defDyanmicVal;
        ArticulationDrive currentDrive = joint.xDrive;
        currentDrive.forceLimit = forceLimit;
        joint.xDrive = currentDrive;
    }
    DisplaySelectedJoint(selectedIndex);
    StoreJointColors(selectedIndex);
}

void SetSelectedJointIndex(int index)
{
    if (articulationChain.Length > 0)
    {
        selectedIndex = (index + articulationChain.Length) % articulationChain.Length;
    }
}

void Update()
{
    bool SelectionInput1 = Input.GetKeyDown("right");
    bool SelectionInput2 = Input.GetKeyDown("left");

    SetSelectedJointIndex(selectedIndex); // to make sure it is in the valid range
    UpdateDirection(selectedIndex);
```

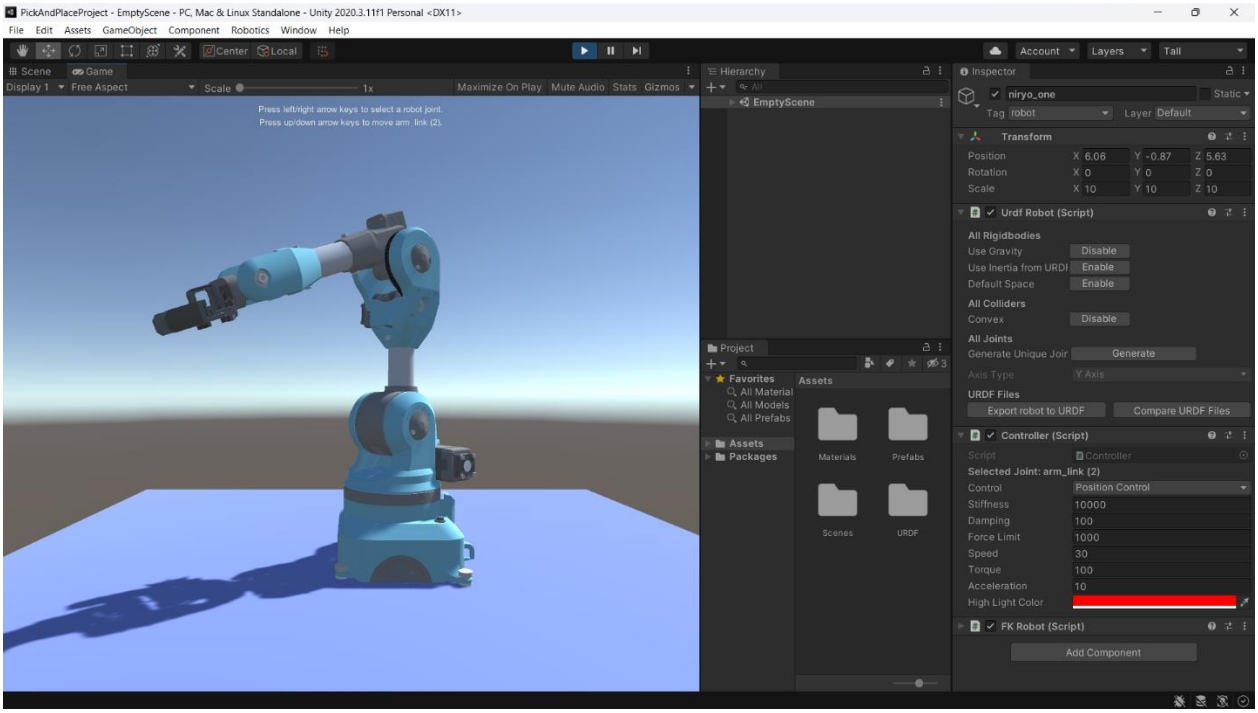
```
        if (SelectionInput2)
            {
                SetSelectedJointIndex(selectedIndex - 1);
                Highlight(selectedIndex);
            }
            else if (SelectionInput1)
            {
                SetSelectedJointIndex(selectedIndex + 1);
                Highlight(selectedIndex);
            }

            UpdateDirection(selectedIndex);
        }
        private void Highlight(int selectedIndex)
        {
            if (selectedIndex == previousIndex || selectedIndex < 0 || selectedIndex >=
articulationChain.Length)
            {
                return;
            }

            // reset colors for the previously selected joint
            ResetJointColors(previousIndex);

            // store colors for the current selected joint
            StoreJointColors(selectedIndex);

            DisplaySelectedJoint(selectedIndex);
            Renderer[] rendererList =
articulationChain[selectedIndex].transform.GetChild(0).GetComponentsInChildren<Renderer>();

            // set the color of the selected join meshes to the highlight color
            foreach (var mesh in rendererList)
            {
                MaterialExtensions.SetMaterialColor(mesh.material, highLightColor);
            }
```

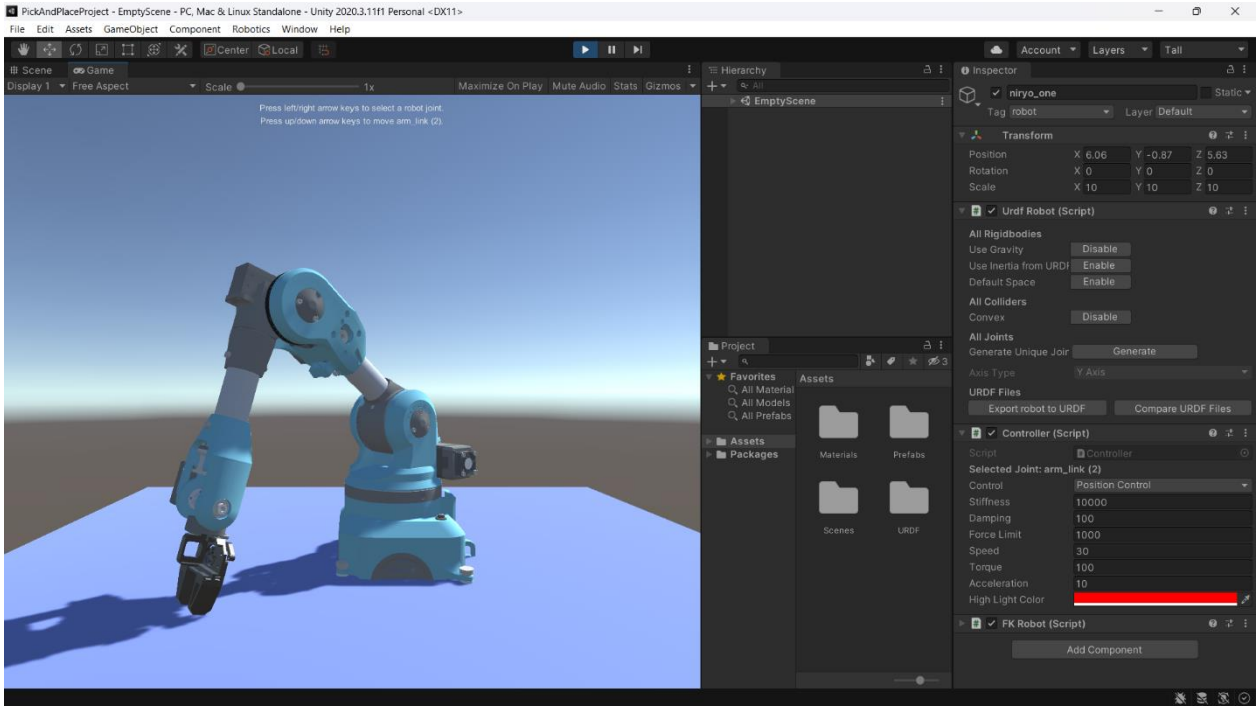
```
}

void DisplaySelectedJoint(int selectedIndex)
{
   if (selectedIndex < 0 || selectedIndex >= articulationChain.Length)
   {
      return;
   }
   selectedJoint = articulationChain[selectedIndex].name + " (" + selectedIndex + ")";
}

private void UpdateDirection(int jointIndex)
{
   if (jointIndex < 0 || jointIndex >= articulationChain.Length)
   {
      return;
   }

   float moveDirection = Input.GetAxis("Vertical");
   JointControl current = articulationChain[jointIndex].GetComponent<JointControl>();
   if (previousIndex != jointIndex)
   {
      JointControl previous = articulationChain[previousIndex].GetComponent<JointControl>();
      previous.direction = RotationDirection.None;
      previousIndex = jointIndex;
   }

   if (current.controltype != control)
   {
      UpdateControlType(current);
   }

   if (moveDirection > 0)
   {
      current.direction = RotationDirection.Positive;
   }
```

```
            else if (moveDirection < 0)
            {
                current.direction = RotationDirection.Negative;
            }
            else
            {
                current.direction = RotationDirection.None;
            }
        }
        private void StoreJointColors(int index)
        {
            Renderer[] materialLists =
articulationChain[index].transform.GetChild(0).GetComponentsInChildren<Renderer>();
            prevColor = new Color[materialLists.Length];
            for (int counter = 0; counter < materialLists.Length; counter++)
            {
                prevColor[counter] = MaterialExtensions.GetMaterialColor(materialLists[counter]);
            }
        }

        private void ResetJointColors(int index)
        {
            Renderer[] previousRendererList =
articulationChain[index].transform.GetChild(0).GetComponentsInChildren<Renderer>();
            for (int counter = 0; counter < previousRendererList.Length; counter++)
            {
                MaterialExtensions.SetMaterialColor(previousRendererList[counter].material,
prevColor[counter]);
            }
        }

        public void UpdateControlType(JointControl joint)
        {
            joint.controltype = control;
            if (control == ControlType.PositionControl)
            {
```

```
            ArticulationDrive drive = joint.joint.xDrive;
            drive.stiffness = stiffness;
            drive.damping = damping;
            joint.joint.xDrive = drive;
        }
    }

    public void OnGUI()
    {
        GUIStyle centeredStyle = GUI.skin.GetStyle("Label");
        centeredStyle.alignment = TextAnchor.UpperCenter;
        GUI.Label(new Rect(Screen.width / 2 - 200, 10, 400, 20), "Press left/right arrow keys to select a robot joint.", centeredStyle);
        GUI.Label(new Rect(Screen.width / 2 - 200, 30, 400, 20), "Press up/down arrow keys to move " + selectedJoint + ".", centeredStyle);
    }
  }
}
```

5.2 OUTPUT SNAPSHOTS





CHAPTER 6: LEARNINGS AND TAKEAWAYS FROM THE STUDY

In my journey of developing a software simulation of a robotic arm in Unity within the context of Digital Twins, I've gleaned invaluable insights and takeaways. Firstly, this experience significantly deepened my understanding of the robotic arm's mechanical intricacies and real-world operations. Unlike static models or traditional documentation, the Digital Twin allowed me to visualize and interact with the arm in a dynamic and immersive manner.

Moreover, the realistic interactions enabled by Unity were enlightening. I came to appreciate how various forces, physics principles, and constraints influence the arm's movements and responsiveness. This understanding proved pivotal in optimizing the arm's performance within the simulation and, by extension, in the physical world.

One of the most significant revelations was the integration of real-time sensor data into the simulation. This process underscored the critical importance of data synchronization for maintaining the Digital Twin's accuracy. It illuminated the potential of Digital Twins in predictive maintenance and performance monitoring for complex systems like robotic arms.

The development journey itself underscored the iterative nature of creating Digital Twins. Regular testing and refinement were essential steps in the process, emphasizing the importance of continuous improvement. This iterative approach allowed me to fine-tune the simulation for a more faithful representation of the robotic arm.

Lastly, I recognized the adaptability of Unity's cross-platform capabilities. This feature allowed me to deploy the simulation on various devices, enhancing its accessibility and usability. It highlighted the importance of creating Digital Twins that can be easily integrated into different operational contexts and workflows.

In conclusion, the experience of developing a software simulation of a robotic arm in Unity for Digital Twin purposes provided a wealth of insights, from a deeper understanding of complex systems to the practical applications of real-time data integration. It reaffirmed the value of iterative development and the adaptability of cross-platform deployment, all contributing to a richer understanding of the potential of Digital Twins in various industries and contexts.

REFERENCES AND ANNEXURES

●www.researchgate.net/publication/337019778_Digital_Twin_Enabling_Technologies_Challenges_and_Open_Research
● https://youtu.be/aWVHR_SyO4c
● https://softengi.com/blog/use-cases-and-applications-of-digital-twin/
● https://research.aimultiple.com/digital-twin-applications/
● https://discover.aveva.com/paid-search-eadt
● https://towardsdatascience.com/digital-twin-with-python-a-hands-on-exa