Republic of the Philippines
Laguna State Polytechnic University
Province of Laguna

College of Computer Studies

CSST 105 - ROBOTICS

FINAL PROJECT: Robotic Arm

Submitted by:

Lesly-Ann B. Victoria
Jonathan Q. Laganzon
Jandel Niño C. Magpantay
Nashrudin Maverick A. Esguerra
Carlo R. Caburnay
John Richard L. Bercades

Submitted to:

Mr. John Peñaredondo

BSCS - IS - 3B

Introduction

Robotic Arm

Robotic Arm are devices that have been designed to do a given activity or job swiftly, correctly, and effectively. They're usually motor-driven consisting of a collection of joints, articulations, and manipulators, and are employed to accomplish heavy and/or repetitive processes quickly and consistently. They are particularly valuable in the industries of industrial production, manufacturing, machining, and assembly ***(Universal Robot, 2022).***

Overview:

This project involves the development of a robotic arm capable of both manual and autonomous operation for picking up and placing objects on a table. The robotic arm integrates advanced perception and decision-making capabilities, allowing it to identify, grasp, and move objects either under human control or autonomously without human intervention. This dual-mode functionality showcases the potential of robotic systems to perform complex tasks in dynamic environments, highlighting both practical applications and technological advancements in the field of robotics. By leveraging state-of-the-art technologies in robotics and AI, this project aims to demonstrate the efficiency, accuracy, and versatility of robotic arms in real-world scenarios.

Objectives:

- **Download and Install the Necessary Application**
- **Set Up the Development Environment**
- **Configure ROS for Autonomous Mode**
- **Understand the Planning Process**
- **Review the Components Used**
- **Analyze the Source Code**
- **View the User Manual**
- **Complete the Assessment**

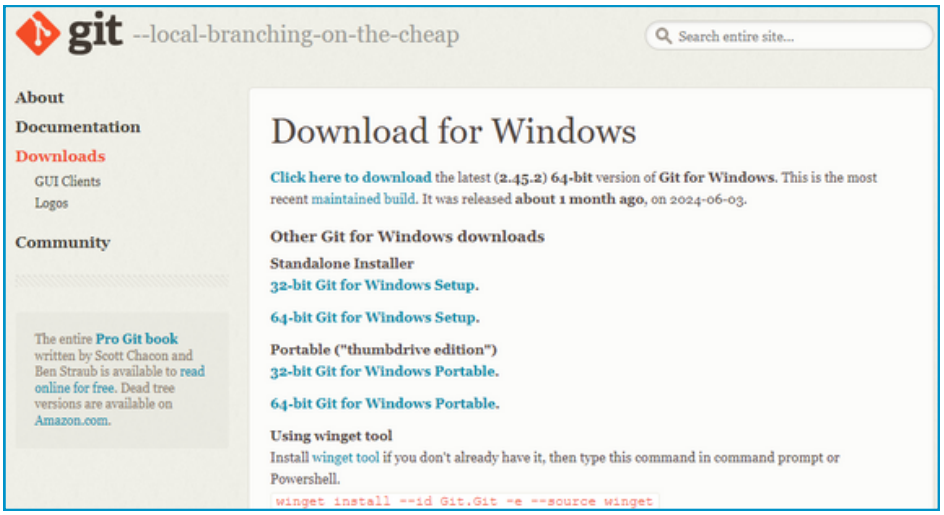
Download Application

Download and Install the Necessary Application:

Git

We used this to build a robotic arm with version control for managing changes and collaboration.
Git a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

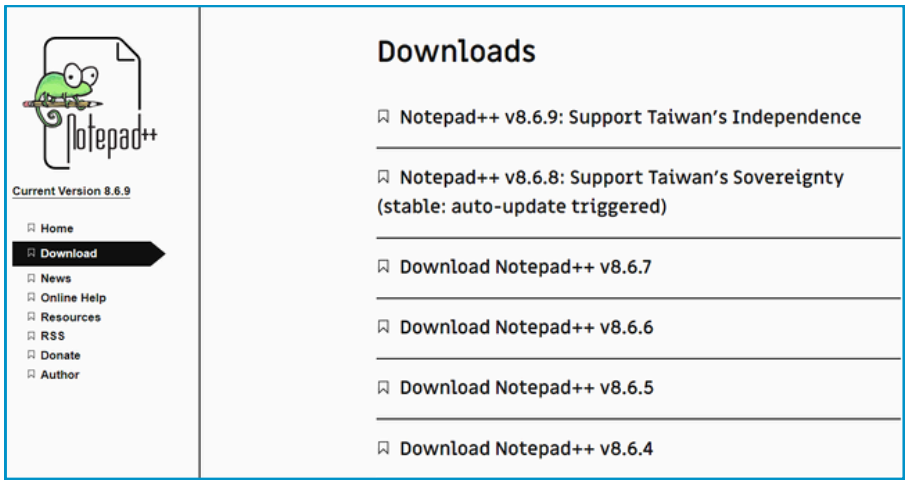
Git Link: https://git-scm.com/download/win



Notepad++

We used this to build a robotic arm with a lightweight text editor for coding.
Notepad++ is a free (as in “free speech” and also as in “free beer”) source code editor and Notepad replacement that supports several languages.

Notepad++ Link: https://notepad-plus-plus.org/downloads/



Download Application

Download and Install the Necessary Application:

Docker Desktop

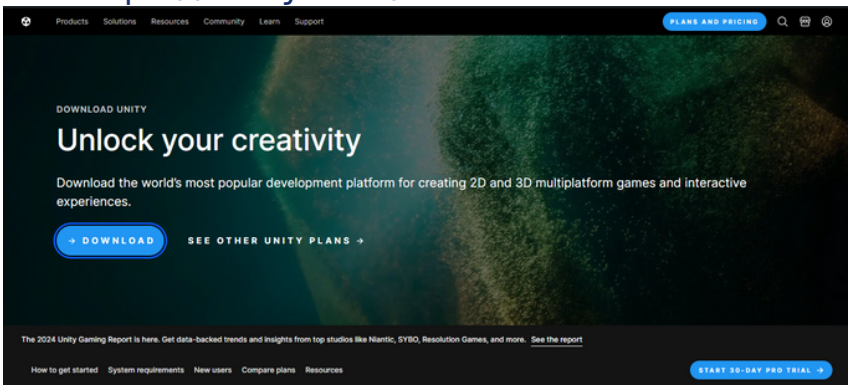
We used this to build a robotic arm with consistent development environments via containers.
Docker Desktop is a one-click-install application for your Mac, Linux, or Windows environment that lets you build, share, and run containerized applications and microservices.

Docker Desktop Link:
https://docs.docker.com/desktop/install/windows-install/

Unity Hub

We used this to build a robotic arm in managing Unity versions and projects for simulations and visualizations.
Unity Hub is a standalone application that streamlines the way you find, download, and manage your Unity Projects and installations. In addition, you can manually add versions of the Editor that you have already installed on your machine to your Hub.

Unity Hub Link: https://unity.com/download



Download Application

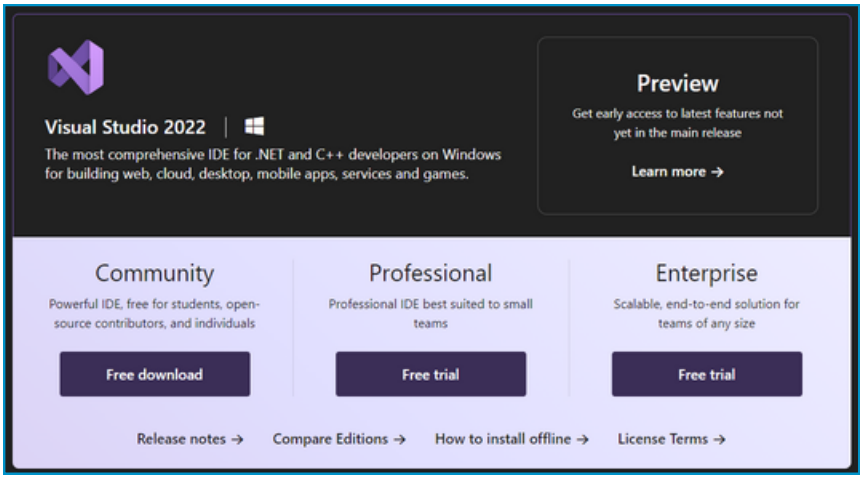
Download and Install the Necessary Application:

Visual Studio 2022

We used this to build a robotic arm for managing Unity versions and projects for simulations and visualizations.
Visual Studio 2022 makes it quick and easy to build modern, cloud-based applications with Azure.

We also used C#

Visual Studio 2022 Link:
https://visualstudio.microsoft.com/downloads/



Download the Robotic Arm Project
Google Drive Link:
https://drive.google.com/drive/folders/1hFK3cdt5VedwPghHEp7no4pRc8wfAxCx?usp=sharing

GitHub Link:
https://github.com/LeslyVictoria2/Robotic-Arm-Project_BSCS3B

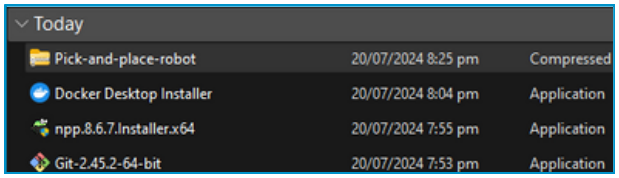
How to download and install the necessary application for the Robotic Arm Project (YouTube Link): https://youtu.be/nZjuVreMQ3c

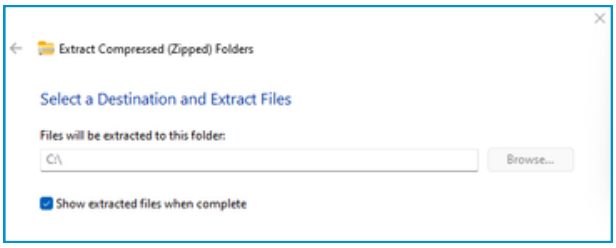
Note:
Download the latest version and this project is only for Windows user.

Environment Setup

Step 1:

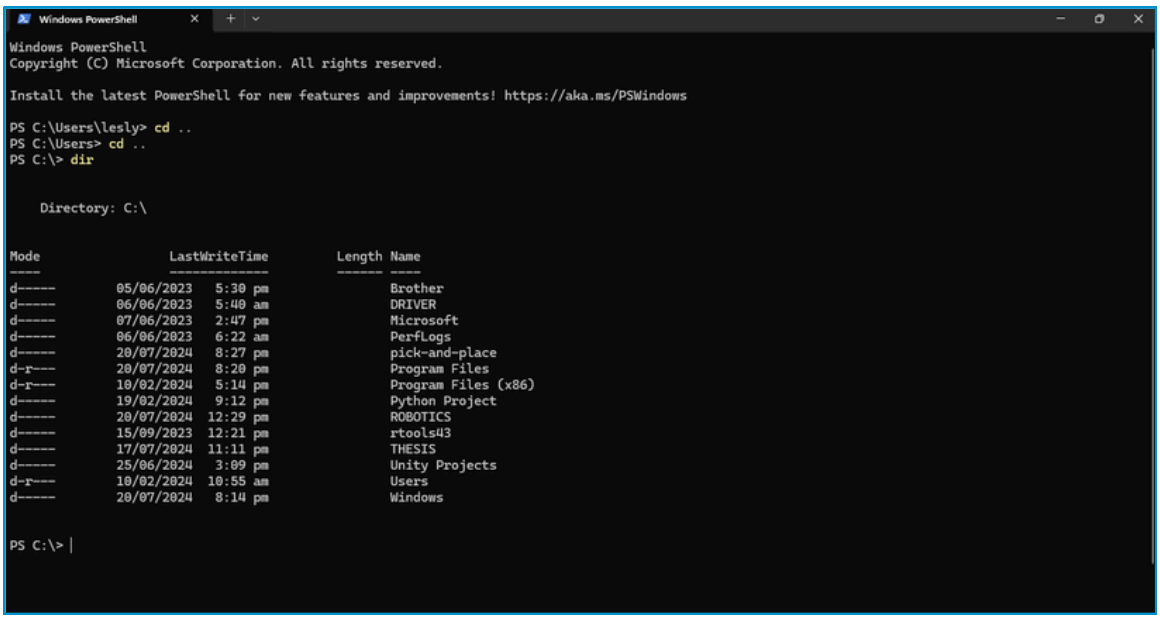
Find the pick-and-place file that you downloaded from the Google Drive Link, then extract the pick-and-place zip file to 'C:\' drive or the main hard drive (SDD) of Windows.





Step 2:

Open your Windows PowerShell, then navigate the project directory,. Type the **'cd ..'** twice to move up two directories, then type **'dir'** to list the contents of the current directory.

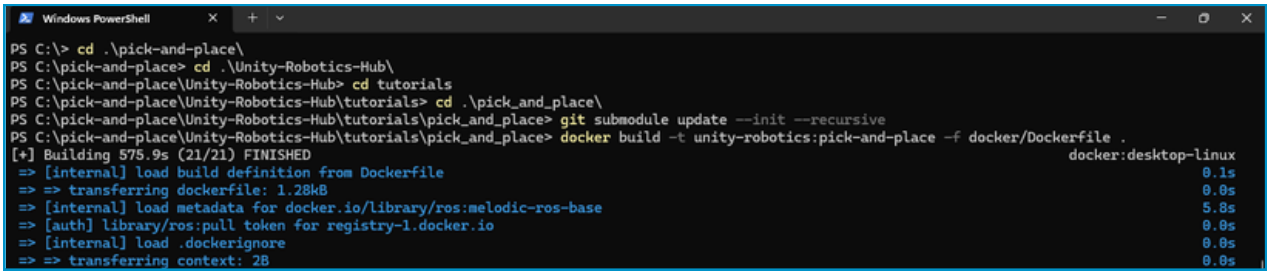


Step 3:

Next use 'cd' change directory commands to navigate the pick_and_place. Type **cd .\pick-and-place** to change the current directory to the pick-and-place folder. Next, type **cd .\Unity-Robotics-Hub** to move into the Unity-Robotics-Hub directory, and then **cd tutorials** to access the tutorials subdirectory. By executing **cd .\pick_and_place**, you will enter the pick_and_place folder.

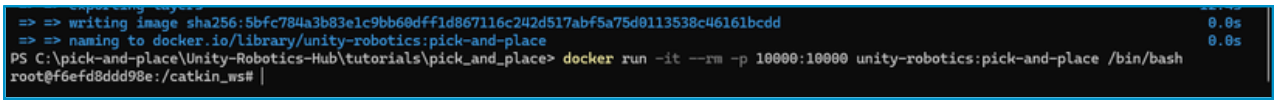
Environment Setup

Type **git submodule update --init --recursive** to initialize and update any git submodules recursively, ensuring all nested repositories are up to date. Finally, build a Docker image using the command **docker build -t unity-robotics:pick-and-place -f docker/Dockerfile .**, which tags the image as unity-robotics:pick-and-place and uses the Dockerfile located in the docker directory to define the build process.



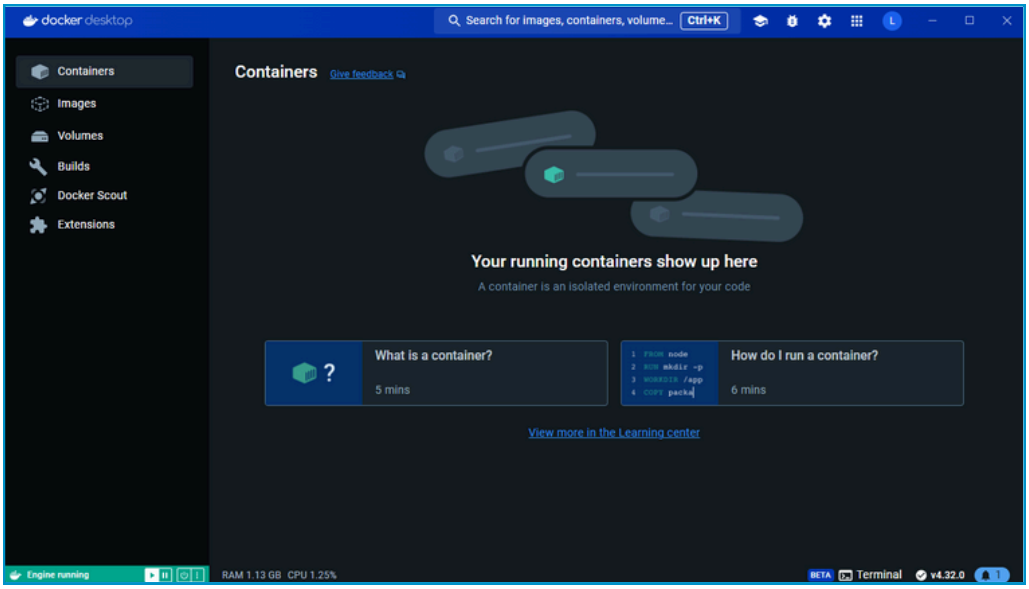
Step 4:

The next step is to run the Docker container by typing `**docker run -it --rm -p 10000:10000 unity-robotics:pick-and-place /bin/bash**`. This command will open the Docker Desktop application. When prompted, click 'Allow access'.



Note:

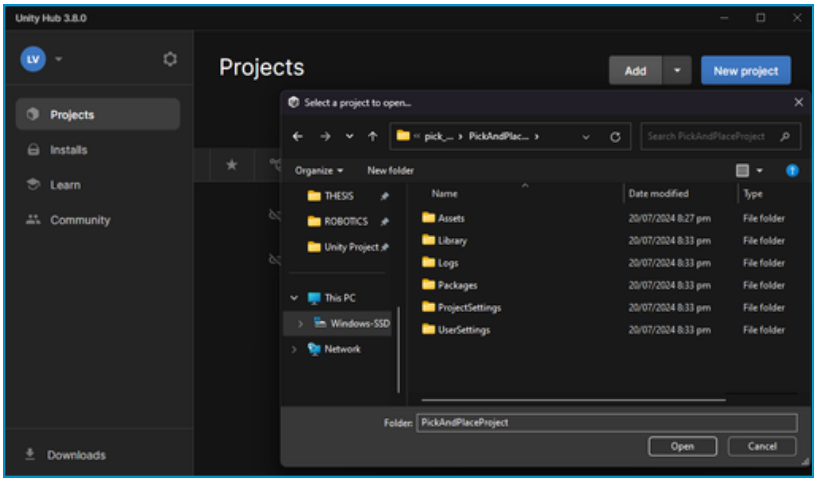
If you encounter an error, you should see if the docker desktop is open to run the code.



Environment Setup

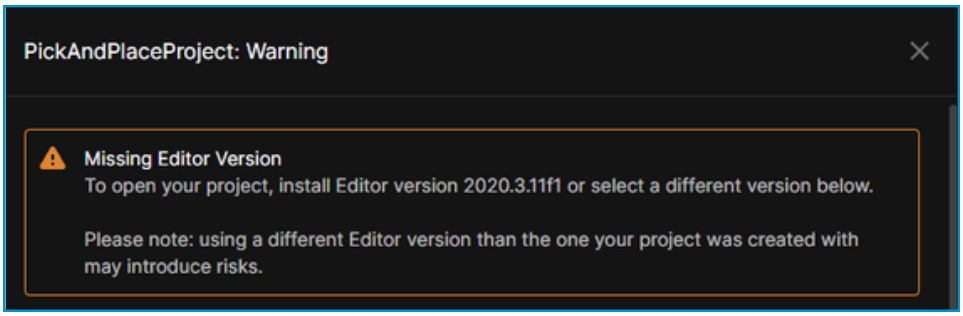
Step 5:

Lastly, open your Unity Hub and press the ‘Add’ button. Navigate to your main hard disk and find the project by following this path: `pick-and-place` > `Unity-Robotics-Hub` > `tutorials` > `pick_and_place` > `PickAndPlaceProject`. This will add the Unity project to your Unity Hub, allowing you to open and work on it.



Note:

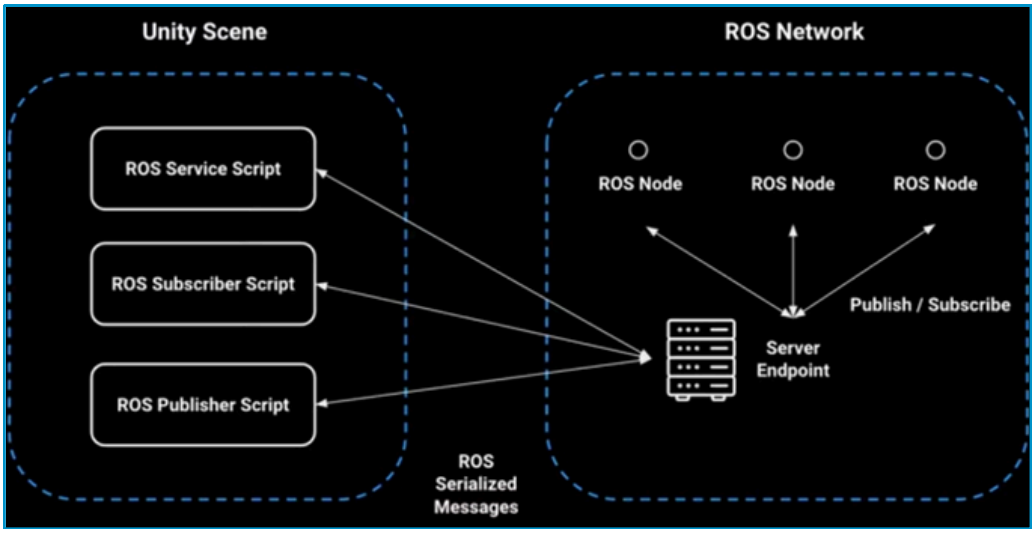
If you get a warning like this, you should install the required Unity Editor version and Visual Studio Community version.



In opening again the project:

- Open Docker Desktop App
- Open Unity Hub App
- Open Windows PowerShell
  - type ‘docker run -it --rm -p 10000:10000 unity-robotics:pick-and-place /bin/bash’
  - type ‘roslaunch niryo_moveit part_3.launch’

Autonomous Mode



Generate ROS Messages:

1. In Unity, look for “Robotics” in the upper left corner of the screen.
2. Select “Generate ROS messages...”.
3. Replace the ROS message path with the correct one by pressing the ‘Browse’ button then type ‘C:/pick-and-place/Unity-Robotics-Hub/tutorials/pick_and_place/ROS’.

Rebuild the Project:

- moveit_msgs:
  - msg
    - RobotTrajectory.msg
- niryo_moveit:
  - msg
    - NiryoMoveitJoints.msg
    - NiryoTrajectory.msg
  - srv

Set Up the Publisher in the Scene:

1. In the Scene tab, under the Hierarchy, look for an object named “Publisher”.
2. Select “Publisher”. The “Inspector” tab will appear on the right side of the screen.
3. Click “Add Component” at the bottom of the Inspector tab.
4. Add the component “Script > Trajectory Planner”.

Autonomous Mode

Assign Elements in Trajectory Planner:

- Drag and drop elements from the Hierarchy to the newly added components in the Inspector:
  - Drag “niryo_one” from the Hierarchy to the “Niryo One” field in the Trajectory Planner.
  - Drag “Target” from the Hierarchy to the “Target” field in the Trajectory Planner.
  - Drag “TargetPlacement” from the Hierarchy to the “Target Placement” field in the Trajectory Planner.

Update Button Event:

Configure OnClick Event:

1. In the Hierarchy tab, find and select “Canvas”.
2. Within the Canvas, select the button you wish to configure.
3. In the Inspector tab, scroll down to the OnClick() event section.
4. Replace the “SourceDestinationPublisher.Publish” event with “Trajectory Planner.PublishJoints”:
   - Click the existing “SourceDestinationPublisher.Publish” to highlight it.
   - Click the dropdown menu and select “Trajectory Planner”.
   - From the next dropdown, select “PublishJoints”.

Run the Autonomous Mode:

roslaunch niryo_moveit part_3.launch

Note:

Ctrl + C - shortcut keys to stop the previous script in Windows PowerShell.

Planning Process

PSEUDOCODE

Manual Mode

```
START

    // Check Shoulder_link
    while true:
        if shoulder_link_moves_to_box_location():
            break
        else:
            continue

    // Check Arm_link
    while true:
        if arm_link_pointed_to_box_location():
            break
        else:
            continue

    // Check Elbow_link
    while true:
        if elbow_link_pointed_to_box_location():
            break
        else:
            continue
    // Check Wrist_link
    while true:
        if wrist_link_pointed_to_box_location():
            break
        else:
            continue

    // Check Left/Right Gripper
    while true:
        if gripper_pointed_to_box_location():
            break
        else:
            continue

    // Grab the object
    grab_object()

    // Move to the desired position
    move_to_desired_position()

END
```

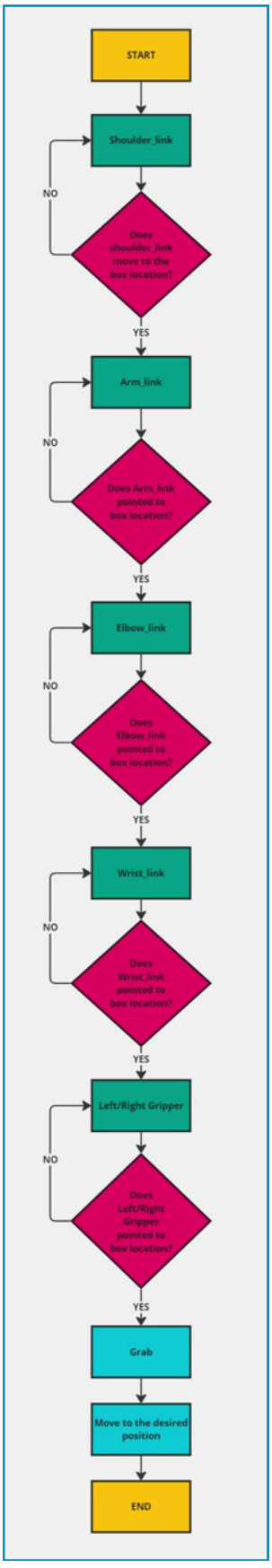
Autonomous Mode

```
START
    // Idle state
    while true:
        if user_pressed_AutonomousMode():
            if object_detected():
                grab_object()
                move_arm_to_desired_location()
                put_object_down()
            else:
                // No object detected, go back to Idle
                continue
        else:
            // User did not press AutonomousMode, go back to Idle
            continue
END
```

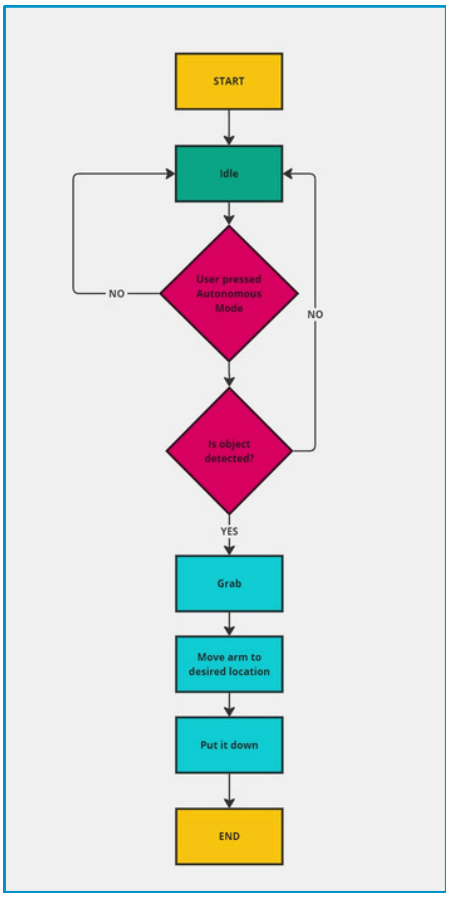
Planning Process

FLOWCHART

Manual Mode

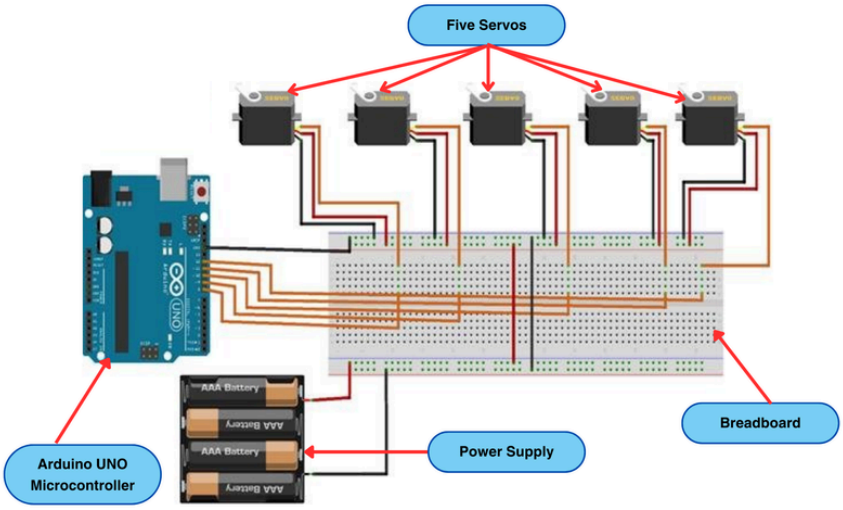


Autonomous Mode



Planning Process

CIRCUIT DIAGRAM



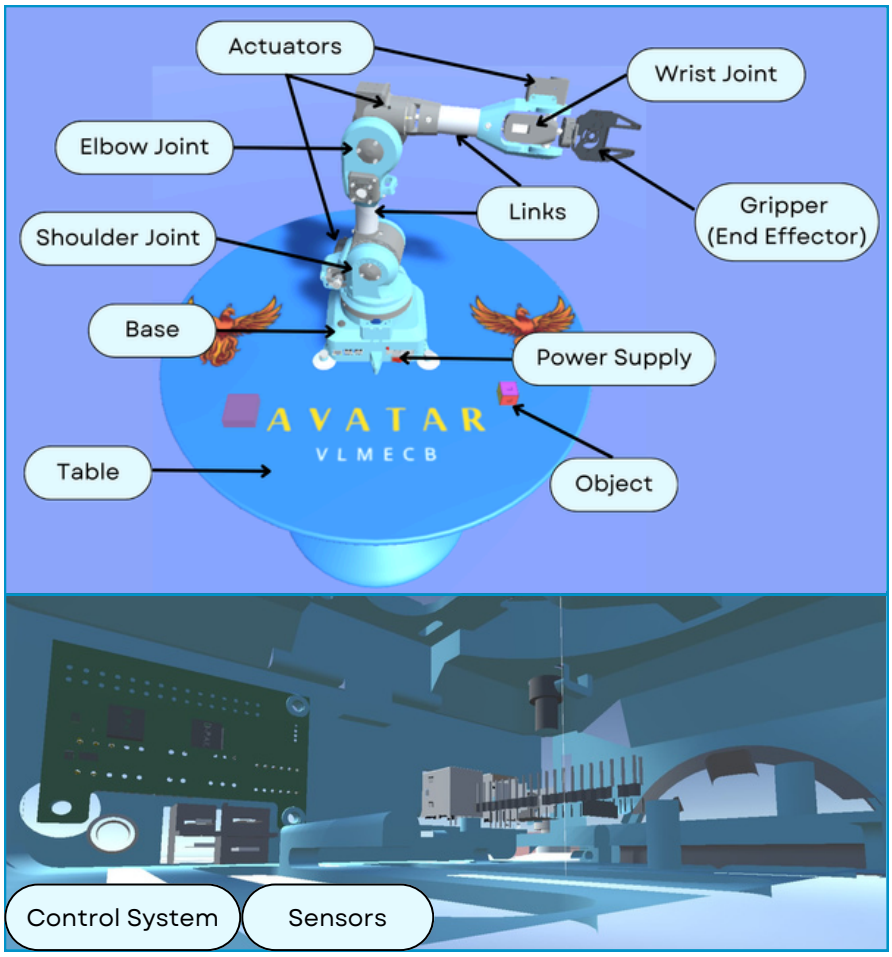
This circuit diagram shows the connections for a robotic arm controlled by an Arduino UNO with manual and autonomous modes, integrating multiple servos and a power supply.

Components of circuit diagram

1. **Arduino UNO Microcontroller:** The main controller for the robotic arm.
2. **Servos:** Five servos.
3. **Breadboard:** Used for organizing and connecting the components.
4. **Power Supply:** AAA battery pack to power the servos and Arduino.

- **Manual Mode:** When the project is opened, the user can move the robotic arm using the keys on the keyboard. Commands are sent to the Arduino via serial communication (e.g., USB connection to a PC running Unity Hub). In Unity, a user interface is created to capture keyboard inputs and send corresponding commands to the Arduino, which controls the servos based on user input.
- **Autonomous Mode:** The user presses the 'Autonomous Mode' button in the Unity interface to switch to autonomous operation. The Arduino then executes pre-programmed sequences stored in its memory. These sequences control the servos to perform specific tasks without further user input.

Robotic Arm Design



Components Used:

1. **Control System**
   - This is the brain of the robotic arm, responsible for processing inputs from sensors and sending commands to the actuators.
   - A microcontroller for low-level control and a computer for high-level tasks and user interface.
2. **Sensors**
   - Sensors collect data about the environment and the state of the robotic arm, such as position, speed, and force.
     - **Encoders** ensure the arm's joints move to precise positions to weld car parts accurately.
     - **Force sensors** help in assembling delicate components like electronic parts without applying too much force.
     - **Vision systems** guide the arm to pick and place components from a conveyor belt, ensuring the correct parts are used.

Robotic Arm Design

- **Proximity sensors** detect nearby obstacles or humans, stopping the arm to prevent accidents.
- **Temperature sensors** monitor the arm's motor and joint temperatures, preventing overheating during continuous operation.

3.**Actuators**

- Actuators convert electrical signals from the control system into physical movement.
- Electric motors at each joint for precise movement.

4.**Joints**

- Joints connect the segments of the robotic arm and allow for movement in various directions.
- A revolute joints providing rotation at the shoulder, elbow, and wrist.

5.**Power Supply**

- The power supply provides the necessary electrical energy for the control system, actuators, and sensors.

6.**End Effector**

- The end effector is the tool attached to the end of the robotic arm that interacts with objects.
- A simple two-fingered gripper for picking and placing objects.

7.**Links**

- Links are the rigid segments that form the structure of the robotic arm.
- Two links (upper arm and forearm) made of lightweight, rigid materials.

8.**Base**

- A heavy, stable platform with an integrated power supply and control unit.

9.**Table**

- A stable surface for mounting the robotic arm and placing objects to be manipulated.

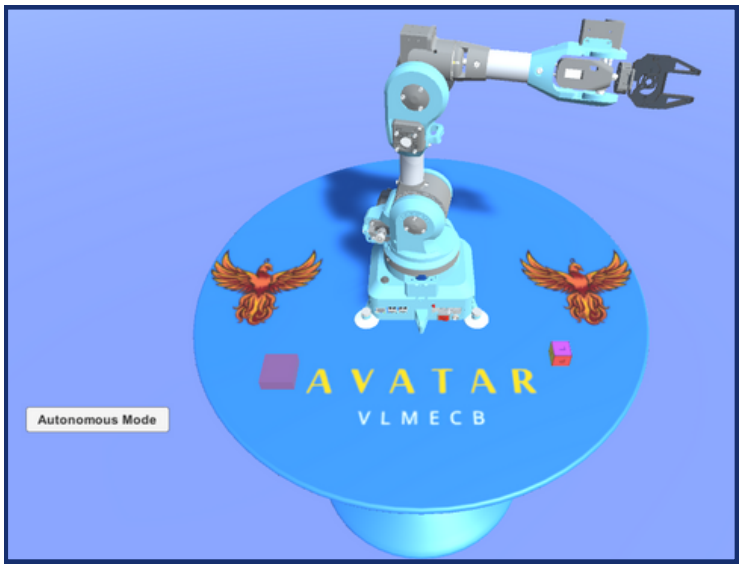
10.**Object**

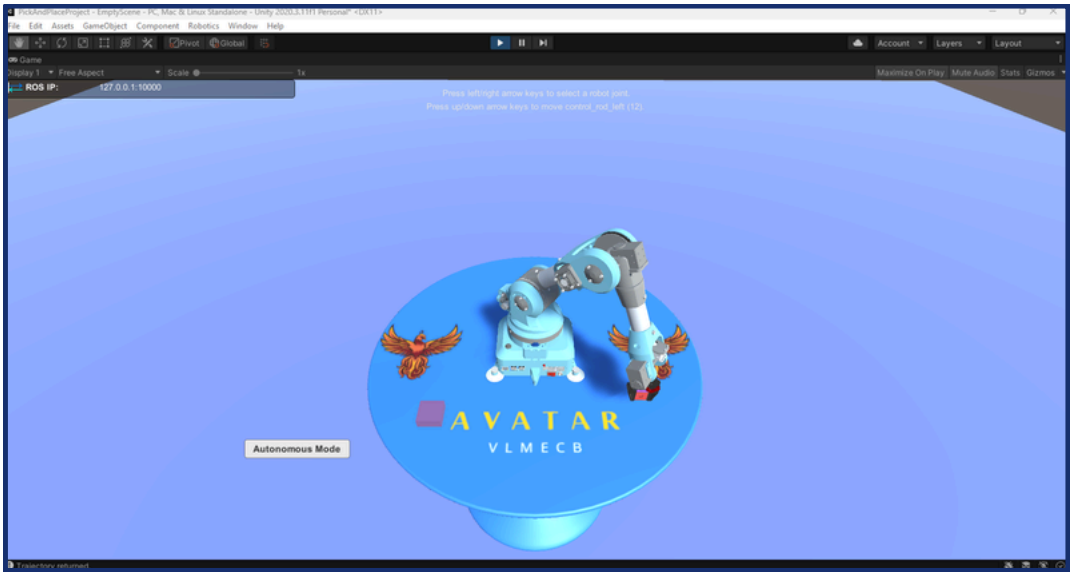
- The item that the robotic arm interacts with during its operation.

Robotic Arm Design

Manual Mode

Autonomous Mode





Source Code

This source code is written in C# and is designed to work with Unity, a popular game engine. We used Visual Studio 2022 as the Integrated Development Environment (IDE) to develop this project. The project integrates with the Robot Operating System (ROS) to control a robotic arm, specifically the Niryo One robot.

Key Components

1. Imports and Namespace:
   - The code imports essential libraries and tools to ensure proper functionality, such as handling collections, string manipulation, and ROS message generation.
2. Class Definition:
   - The main class, RobotTrajectoryMsg, describes the robot's movement trajectory. It includes fields for joint movements and constructors for initializing these fields.
3. Serialization and Deserialization:
   - Deserialization converts data from a message format back into an object, while serialization does the reverse, preparing objects to be sent as messages.
4. ToString Method:
   - This method provides a readable summary of the object's contents, useful for debugging and logging.
5. Registration:
   - Ensures that the RobotTrajectoryMsg class is registered within the system, similar to adding a new employee to a company directory.

This source code is essential for controlling and monitoring the movements of a robotic arm using Unity and ROS, providing detailed control over the robot's actions.

Google drive Link for Source Code:
https://drive.google.com/drive/folders/1NaakVuSWVbsaNvRCDXEYlf0gl_-O-8X5?usp=sharing

User Manual

Control the Robotic Arm:
There are two ways to control the robotic arm manually and autonomously.

Manual Mode:

- Users need to follow a series of steps involving keyboard inputs to control the robotic arm.

Step 1:
Select a Component Press the left or right arrow keys to cycle through the available components (robot joint, actuator, link, or end effector).
Note: The base of the robotic arm can be selected but not moved.
Step 2:
Move the Selected Component Once the desired component is selected, press the up or down arrow keys to move it accordingly.
Step 3:
Pick Up an Object Select the gripper using the left or right arrow keys. Move the gripper to the object by pressing the up or down arrow keys. Position the gripper around the object to grip it.
Step 4:
Place the Object on the Table Follow the steps to select the appropriate component (Step 1) and move the robotic arm (Step 2) to position it above the table. Select the gripper again using the left or right arrow keys. Move the gripper to release the object on the table by pressing the appropriate key.

Autonomous Mode:

- Users will click the "***Autonomous Mode***" button, and the robotic arm will move on its own based on its programming. The robotic arm uses different sensors to track and pick up objects, then place them on the table without further user intervention.

Assessment

Quizzes and Answer Key:

- **Robotic Arm Quiz 1 Link:** https://forms.gle/ne3W5yeYQXg3qD6dA
- **Identification and True or False**

Answer Key (15ITEMS):

1. ROBOTIC ARM
2. ARDUINO UNO MICROCONTOLLER or ARDUINO
3. BREADBOARD
4. MANUAL MODE AND AUTONOMOUS MODE
5. JOINTS, ACTUATORS, END EFFECTORS, POWER SUPPLY, CONTROL SYSTEM
6. FALSE
7. FALSE
8. TRUE
9. TRUE
10. FALSE

- **Robotic Arm Quiz 2 Link:** https://forms.gle/BncFYfWJ7WwdEGg96
- **Fill in the Blank**

Answer Key (10ITEMS):

1. ROBOT OPERATING SYSTEM
2. LINKS
3. C#
4. cd
5. WINDOWS POWERSHELL
6. ZIP FILE
7. dir
8. Git
9. GRIPPER
10. DOCKER DESKTOP

Assessment

Quiz and Answer Key:

- **Robotic Arm Quiz 3 Link:** https://forms.gle/7mT8C6gUyXerNXDt9
- **Identify if the code is in manual mode or autonomous mode.**

Answer Key (10ITEMS - 30POINTS):

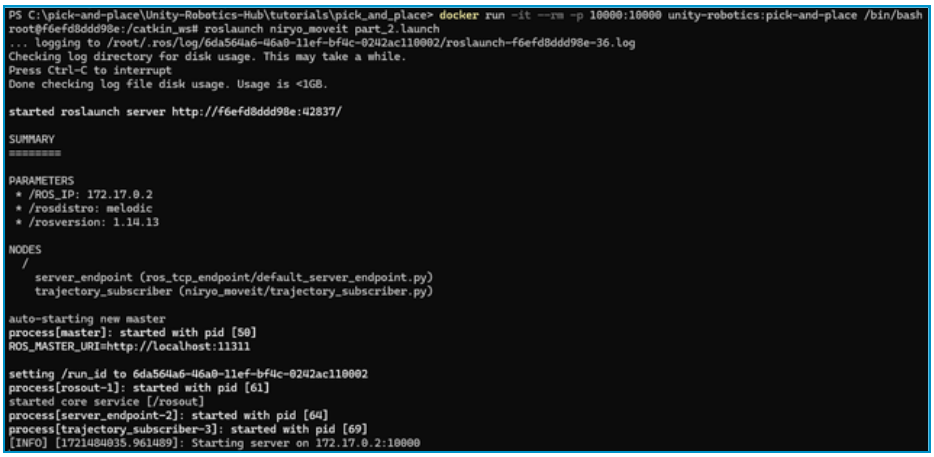
1. Manual Mode
2. Autonomous Mode
3. Autonomous Mode
4. Manual Mode
5. Manual Mode
6. Manual Mode
7. Autonomous Mode
8. Autonomous Mode
9. Autonomous Mode
10. Manual Mode

Assignment:

- **Essay**
- In 2 paragraph, explain the components and functions of a robotic arm, explore its applications in various fields, and discuss the potential advancements and challenges in robotic arm technology.

Practical Task:

- Run the Autonomous Mode
- Hint: Once you have finished downloading the necessary apps and setting up your environment, read and understand the autonomous mode to be able to use it.



Conclusion

The development of a robotic arm capable of both manual and autonomous operation marks a significant advancement in the field of robotics. Such a dual-mode system showcases the flexibility and adaptability of robotic technology in various environments, highlighting its potential to transform industries ranging from manufacturing to service sectors. The importance of this project lies in its ability to demonstrate the practical applications of robotics in performing complex tasks with high efficiency and accuracy, which can lead to increased productivity and safety in workplaces by reducing the need for human intervention in hazardous or repetitive tasks.

Upon completing this project, participants will have gained a comprehensive understanding of both manual and autonomous operation modes of a robotic arm. They will have hands-on experience with the setup and configuration of the development environment, including the installation of necessary applications and configuring the Robot Operating System (ROS) for autonomous operations. This practical knowledge will extend to the planning process, where they will learn to design and implement the steps required for the robotic arm to perform tasks, including object identification, grasping, and placement. Moreover, participants will be well-versed in analyzing source code and understanding the integration of advanced perception and decision-making algorithms in robotics.