**Lab Workbook**

Logo

Description automatically generated

Faculty Name: **Dr. Shraddha Arora**

Student Name: **Piyush Gambhir**

Roll No.: **21CSU349**

Semester: **VI**

Group: **AIM-B (A3)**

**Department of Computer Science and Engineering**

**The NorthCap University**

**Gurugram- 122017, India**

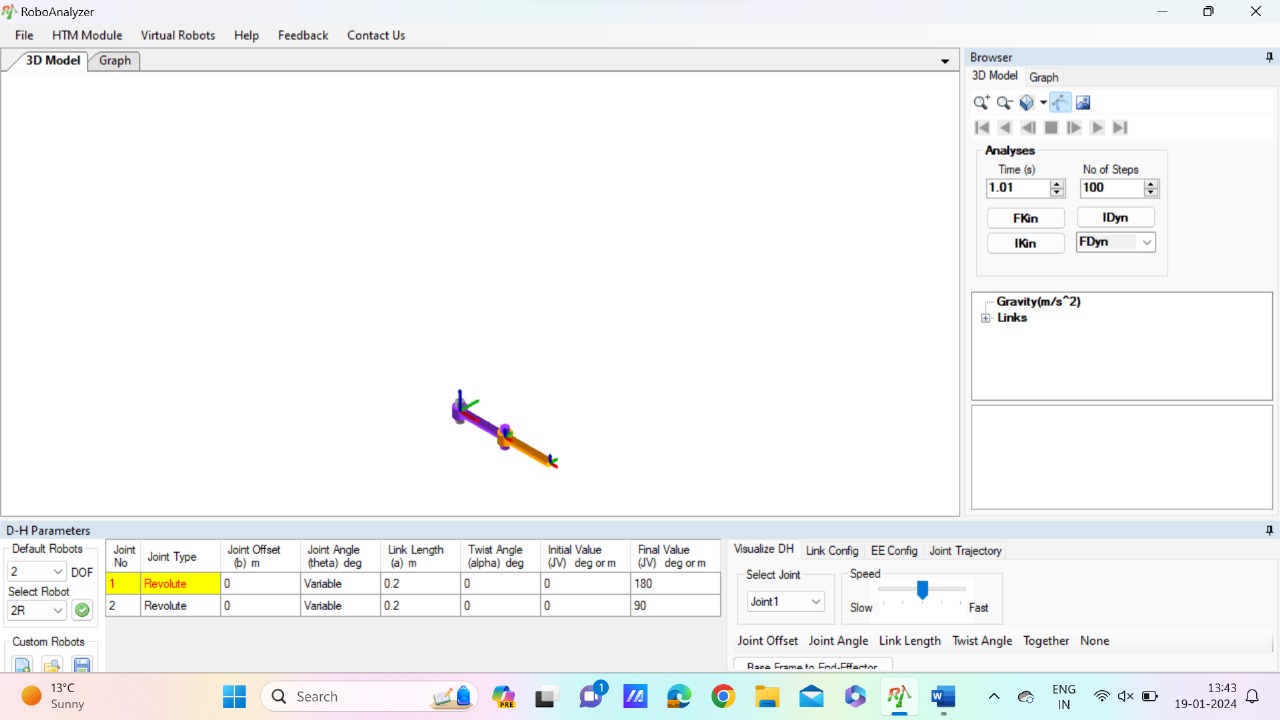
**Session 2023-24**

**Experiment No: 1**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section**: 6th Semester – AIML-B (A3) |
| **Link to Code:** <https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/tree/main/NCU-CSL349%20-%20AI%20For%20Robotics%20-%20Lab%20Manual> |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective(s):**  To get familiarized with the Robo Analyzer stimulator and its features |
| **Outcome:**  Students will learn the physics of robotics with the joy of RoboAnalyzer before attempting to learn the mathematics of robots  Problem Statement :  • To get familiarized with RoboAnalyzer and its basics  • Study the RoboAnalyzer Software, its features and its functionality |

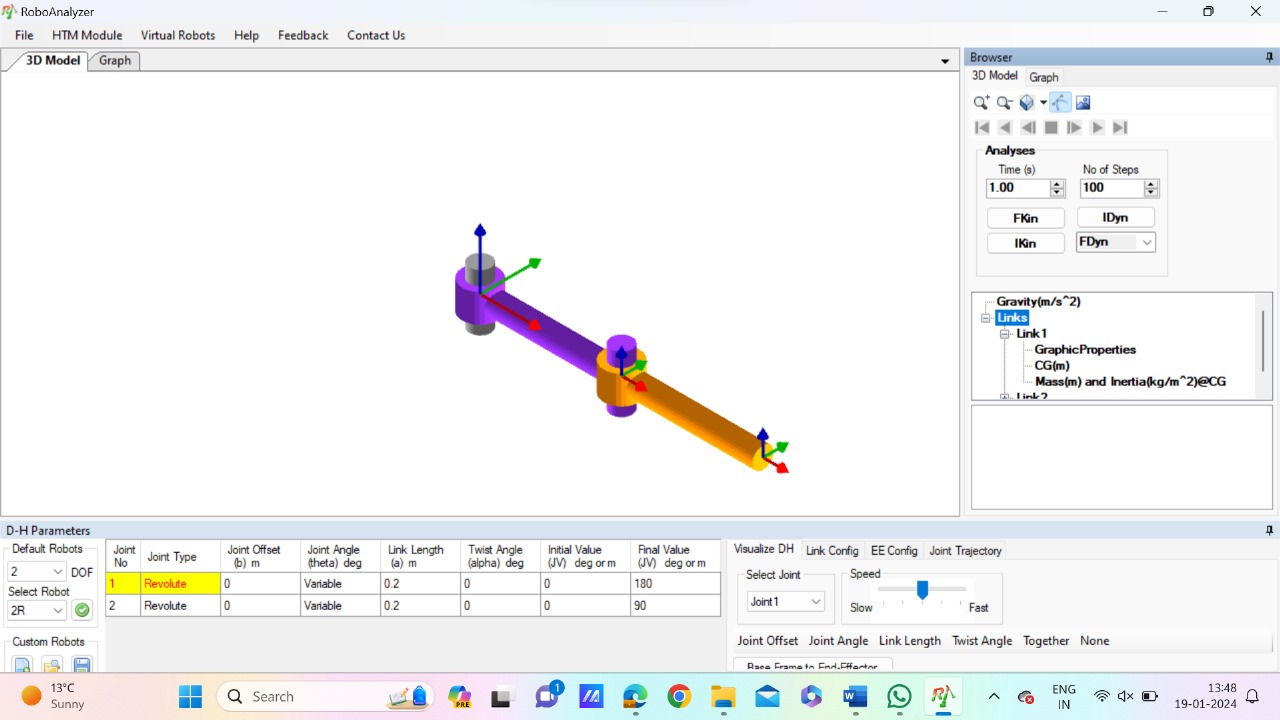
**Student Work Area**



The highlighted object in the image is a 3D model of a two-link robot arm. It is a type of serial manipulator, which means that it consists of two links connected by a single joint. The first link is fixed to the base, and the second link is connected to the first link by a revolute joint. This type of robot arm is commonly used in industrial applications, such as pick-and-place tasks and assembly.

Here are some of the key features of the 3D model in the image:

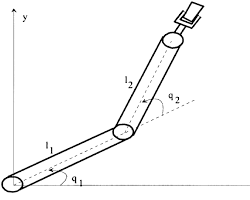
* Two links: The robot arm has two links, which are the upper arm and the lower arm.
* Revolute joint: The two links are connected by a revolute joint, which allows the lower arm to rotate about the upper arm.
* End effector: The end effector is the tool or gripper that is attached to the end of the lower arm. The image does not show the end effector, but it is typically used to grasp and manipulate objects.
* Degrees of freedom: The robot arm has two degrees of freedom, which means that it can be controlled by two independent motors. The first motor controls the rotation of the upper arm, and the second motor controls the rotation of the lower arm.

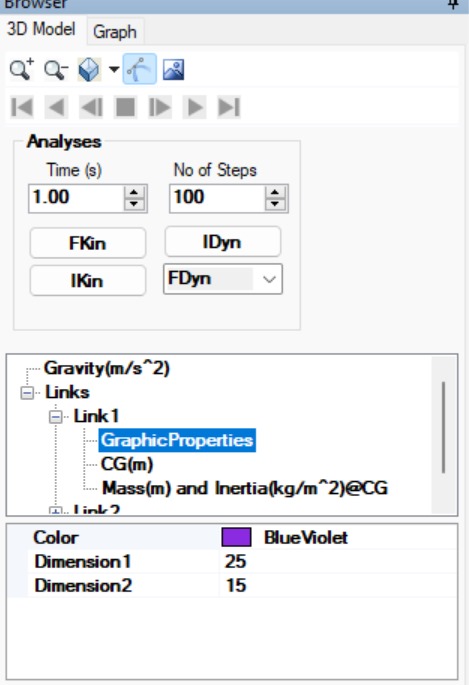


* Two-link robot arm: This is the most general term for this type of robot arm, which has two links connected by a single joint.
* Serial manipulator: This term refers to the fact that the links are connected in a chain, with each link moving relative to the one before it.
* Revolute joint: This type of joint allows the two links to rotate about each other.
* Link 1: This is the first link of the robot arm, which is fixed to the base.
* Link 2: This is the second link of the robot arm, which is connected to Link 1 by the revolute joint.
* End effector: This is the tool or gripper that is attached to the end of the lower arm. The image does not show the end effector, but it is typically used to grasp and manipulate objects.
* Degrees of freedom: The robot arm has two degrees of freedom, which means that it can be controlled by two independent motors.

Here are some additional keywords that you could use to describe the robot arm, depending on the context:

* Anthropomorphic: This term is used to describe robot arms that resemble the human arm, with a shoulder, elbow, and wrist.
* Cylindrical: This term is used to describe robot arms that have a cylindrical workspace, in which the end effector can move up and down and around in a circle.
* SCARA: This is a type of robot arm that is commonly used in pick-and-place tasks. It has two revolute joints and one prismatic joint, which allows the end effector to move in a square pattern.

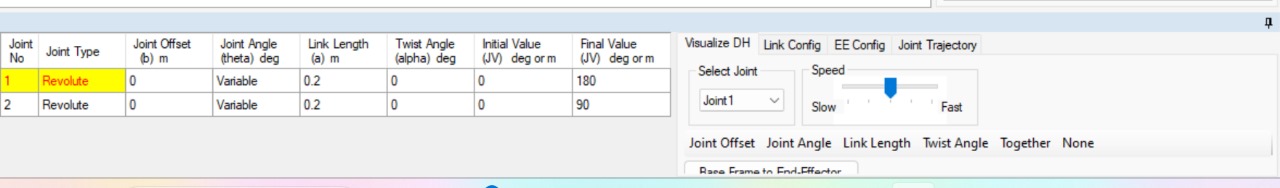
[](https://www.researchgate.net/figure/Two-link-Planar-Robot-Arm_fig1_231914122)



* Force cube: This is the central object in the image, and it is labeled as such. It is likely a tool used to measure or apply forces in 3D modeling or simulation software.
* FKin and IDyn: These are likely labels for the forward kinematics and inverse dynamics of the robot arm. Forward kinematics calculates the position and orientation of the end effector based on the joint angles, while inverse dynamics calculates the joint torques required to achieve a desired end effector position and orientation.
* IKin and FDyn: These are likely labels for the inverse kinematics and forward dynamics of the robot arm. Inverse kinematics calculates the joint angles required to achieve a desired end effector position and orientation, while forward dynamics calculates the motion of the robot arm based on the applied forces and torques.
* Gravity(m/s^2): This label indicates that the simulation is taking gravity into account, and the value specifies the acceleration due to gravity.
* Links: These are the individual components of the robot arm, which are connected by joints. The image shows two links, but there may be more depending on the specific robot arm model.
* Joint: This is the point at which two links are connected. The image shows a revolute joint, which allows the two links to rotate about each other.
* CG(m): This likely refers to the center of gravity of the robot arm.
* Mass(kg) and Inertia(kg/m^2)@CG: These labels indicate the mass and inertia of the robot arm at its center of gravity. These values are important for calculating the robot arm's motion.

Here are some additional keywords that you could use to describe the image, depending on the context:

* Robot arm: This is the general term for the mechanical manipulator shown in the image.
* 3D model: The image is a 3D model of a robot arm, which means that it is a computer-generated representation of the robot arm's geometry.
* Simulation: The image is likely from a simulation of the robot arm, which means that it is being used to study the robot arm's behavior under different conditions.



* Software interface: The main part of the image shows a user interface of a software program. It has menus, toolbars, and panels that allow users to interact with and manipulate 3D models.
* Robot arm: The 3D model in the center of the image is a six-axis robot arm. It has six joints that allow it to move in different directions. The specific type of robot arm is difficult to determine from the image, but it appears to be a common industrial robot design.
* Robot arm components: The different parts of the robot arm are labeled in the image, including the base, links, joints, and end effector. The end effector is the gripper or tool attached to the end of the robot arm, and it is used to interact with objects in the environment.
* Motion planning: The software program appears to be designed for motion planning, which is the process of creating a sequence of movements for the robot arm to achieve a desired task. The panels on the right side of the image likely show options for planning and controlling the robot arm's movements.
* Simulation: The 3D model and the software interface suggest that this is a simulation of a robot arm. This means that the robot arm's behavior can be modeled and tested in a virtual environment before it is deployed in the real world.

Here are some additional keywords that you could use to describe the image, depending on the context:

* Industrial robot: The robot arm in the image is likely used in an industrial setting, such as for manufacturing or assembly tasks.
* Computer-aided design (CAD): The software program may be part of a CAD system, which is used to design and create 3D models of objects.
* Robotics: The image is related to the field of robotics, which is the study and design of robots.

**Experiment No: 2**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section**: 6th Semester – AIML-B (A3) |
| **Link to Code:** <https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/tree/main/NCU-CSL349%20-%20AI%20For%20Robotics%20-%20Lab%20Manual> |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective(s):**  TO GET FAMILIARIZE WITH THE BASICS OF MATLAB |

**Student Work Area**

**MATLAB Lab Manual for Beginners**

**1. Introduction to MATLAB**

* **Overview:**
  + MATLAB is a high-performance computational software environment for numerical computation, visualization, programming, and more.
  + Widely used in science, engineering, mathematics, finance, and other fields.
* **Basic Operations:**
  + Starting MATLAB: Double-click the MATLAB icon.
  + Command Prompt: Enter commands, view results, and interact with MATLAB.
  + Workspace: Stores variables, arrays, and other data during a session.
  + Help System: Extensive documentation and examples (click Help menu).

**2. Variables and Data Types**

* **Variables:**
  + Named containers for storing values.
  + Create using assignment statements (e.g., x = 10).
  + Data types:
    - Numeric (integers, floating-point, complex)
    - Character (strings)
    - Logical (true or false)
    - Cell arrays (flexible collections of different data types)
* **Arrays:**
  + Ordered collections of elements of the same data type.
  + Vectors (one-dimensional) and matrices (two-dimensional).
  + Creating arrays:
    - Colon operator (e.g., 1:10 for integers 1 to 10).
    - Linear spacing (linspace()).
    - Random numbers (rand()).
    - Built-in functions (e.g., zeros(), ones(), diag()).
  + Indexing and accessing elements (e.g., x(2) for the second element).

A screenshot of a computer

Description automatically generated

**3. Operators and Expressions**

* **Arithmetic Operators:**
  + +, -, \*, /, ^ (for exponentiation)
  + Precedence rules (PEMDAS).
* **Relational Operators:**
  + ==, ~=, >, <, >=, <=
  + Used for logical comparisons.
* **Logical Operators:**
  + &, |, ~ (logical AND, OR, NOT)
  + Combine logical expressions.
* **Assignment Operators:**
  + =, +=, -=, \*=, /=, %= (modulus)
  + Assign values and perform operations simultaneously.

A screenshot of a computer

Description automatically generated

**4. Control Flow**

* **Conditional Statements:**
  + if, else if, else, end
  + Execute code based on conditions.
* **Loops:**
  + for, while, break, continue
  + Repeat code blocks multiple times.

**5. Functions**

* **Built-in Functions:**
  + MATLAB provides a wide range of functions for mathematical operations, data analysis, plotting, and more.
  + Access help using help function\_name.
* **User-Defined Functions:**
  + Create your own functions using function keyword.
  + Encapsulate code for reusability and modularity.

**6. Script Files and M-Files**

* **Script Files (.m files):**
  + Sequence of MATLAB commands saved as a text file.
  + Execute by running from the Command Prompt.
* **M-Files:**
  + Can include functions, variables, and other MATLAB constructs.
  + Organize code, improve readability, and make it reusable.

**7. Input and Output**

* **User Input:**
  + Use input() function to prompt the user for input.
* **File I/O:**
  + Read and write data to text files (dlmwrite(), dbread()).
  + Load and save MATLAB variables (save(), load()).

**8. Plotting and Visualization**

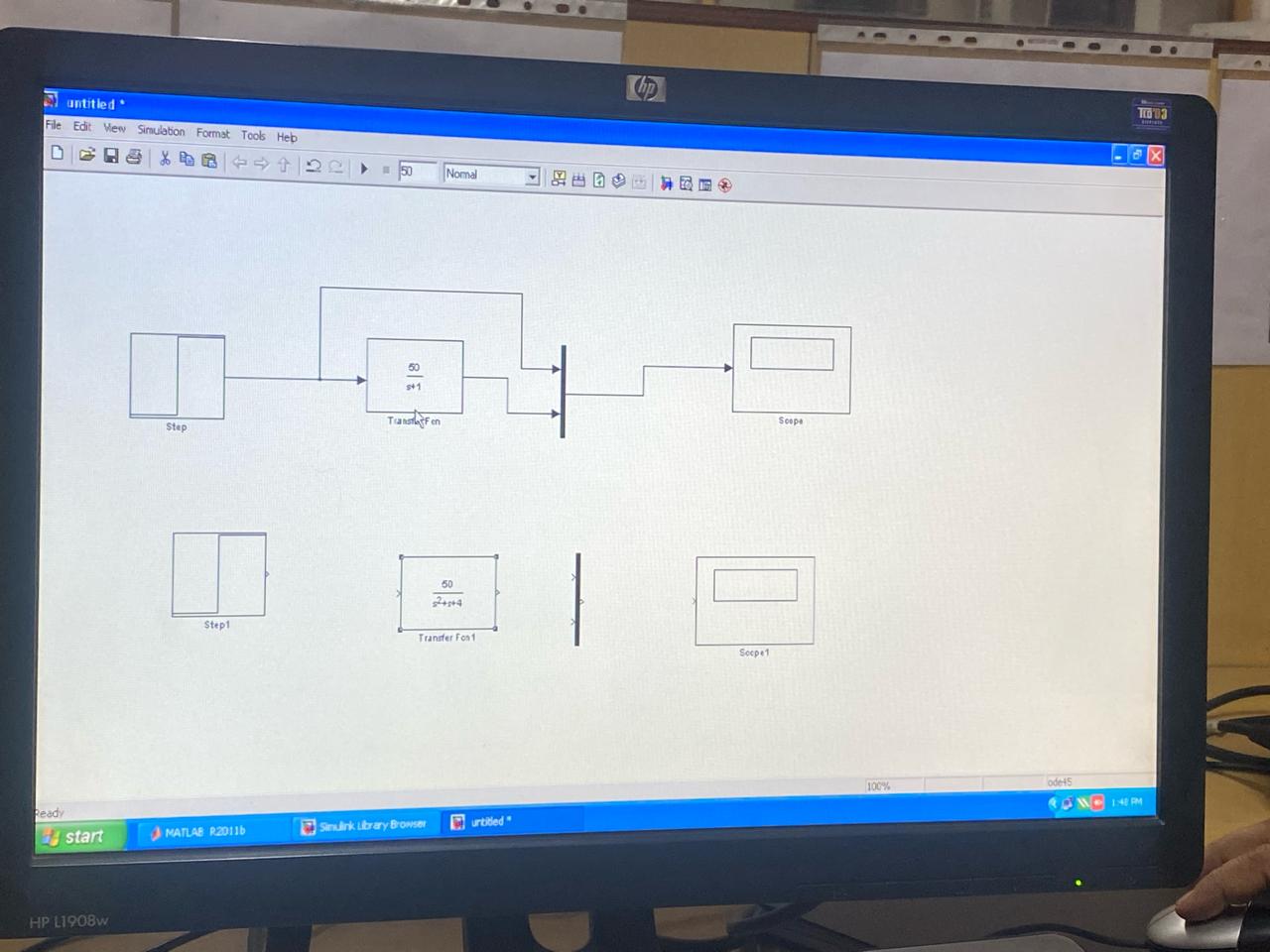
* **Basic Plots:**
  + plot(), bar(), stem(), pie() for various plot types.
  + Customize lines, markers, colors, axes labels, and titles.
* **Advanced Visualization:**
  + subplot(), imagesc(), contour(), surf() for more complex plots.
  + 3D visualization with toolboxes like Graphics.

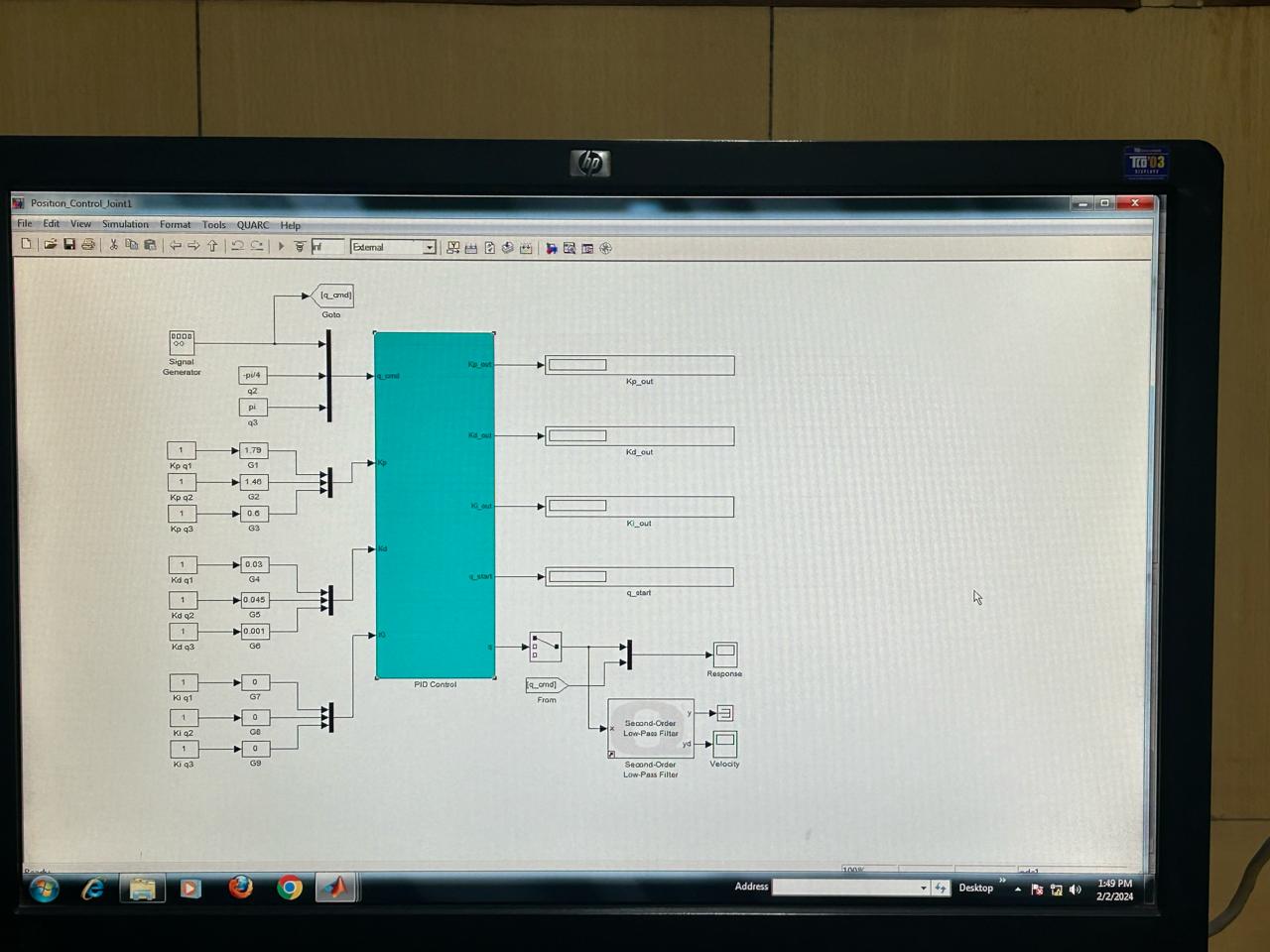
**Experiment No: 3**

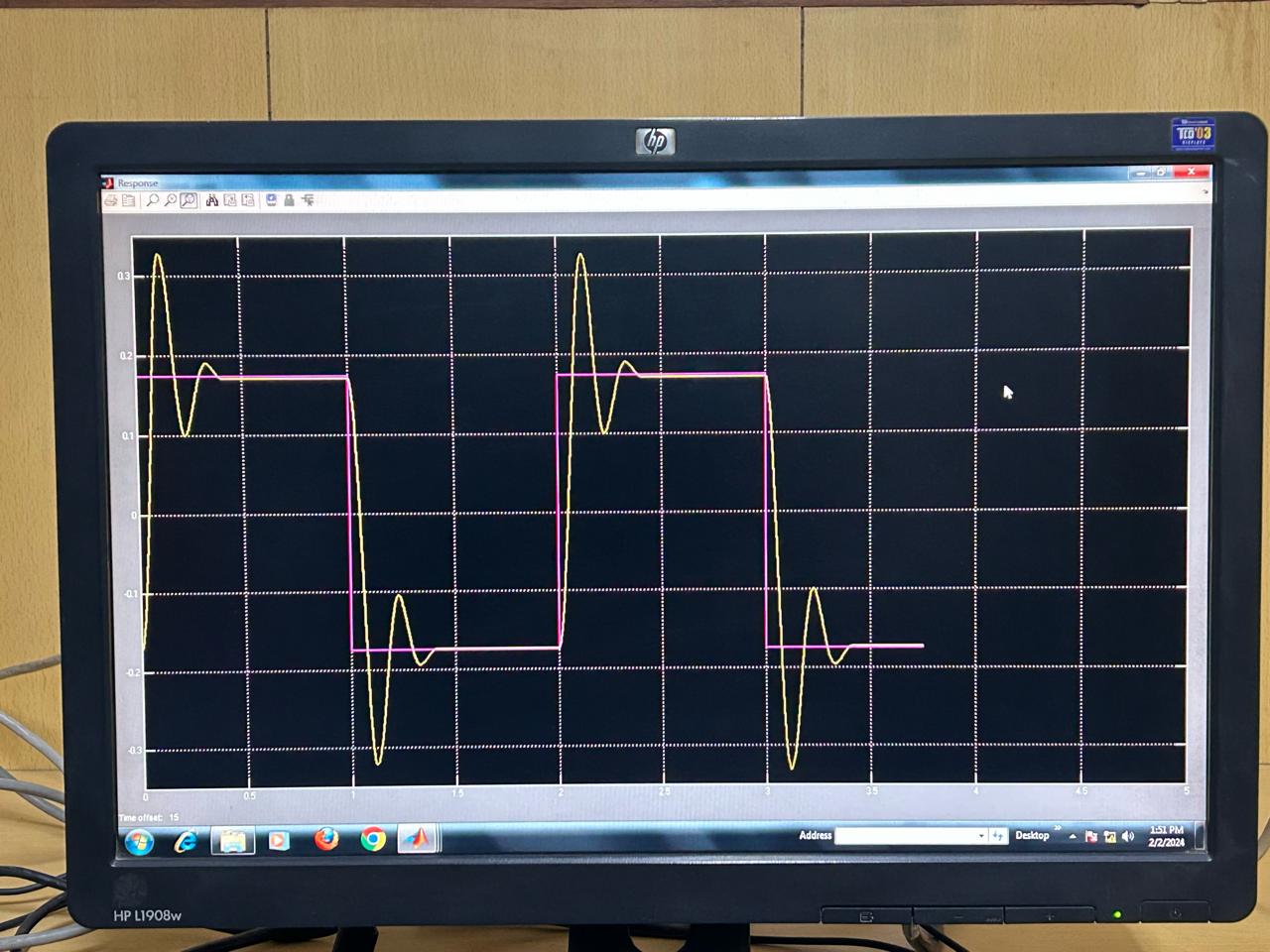
|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section**: 6th Semester – AIML-B (A3) |
| **Link to Code:** <https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/tree/main/NCU-CSL349%20-%20AI%20For%20Robotics%20-%20Lab%20Manual> |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

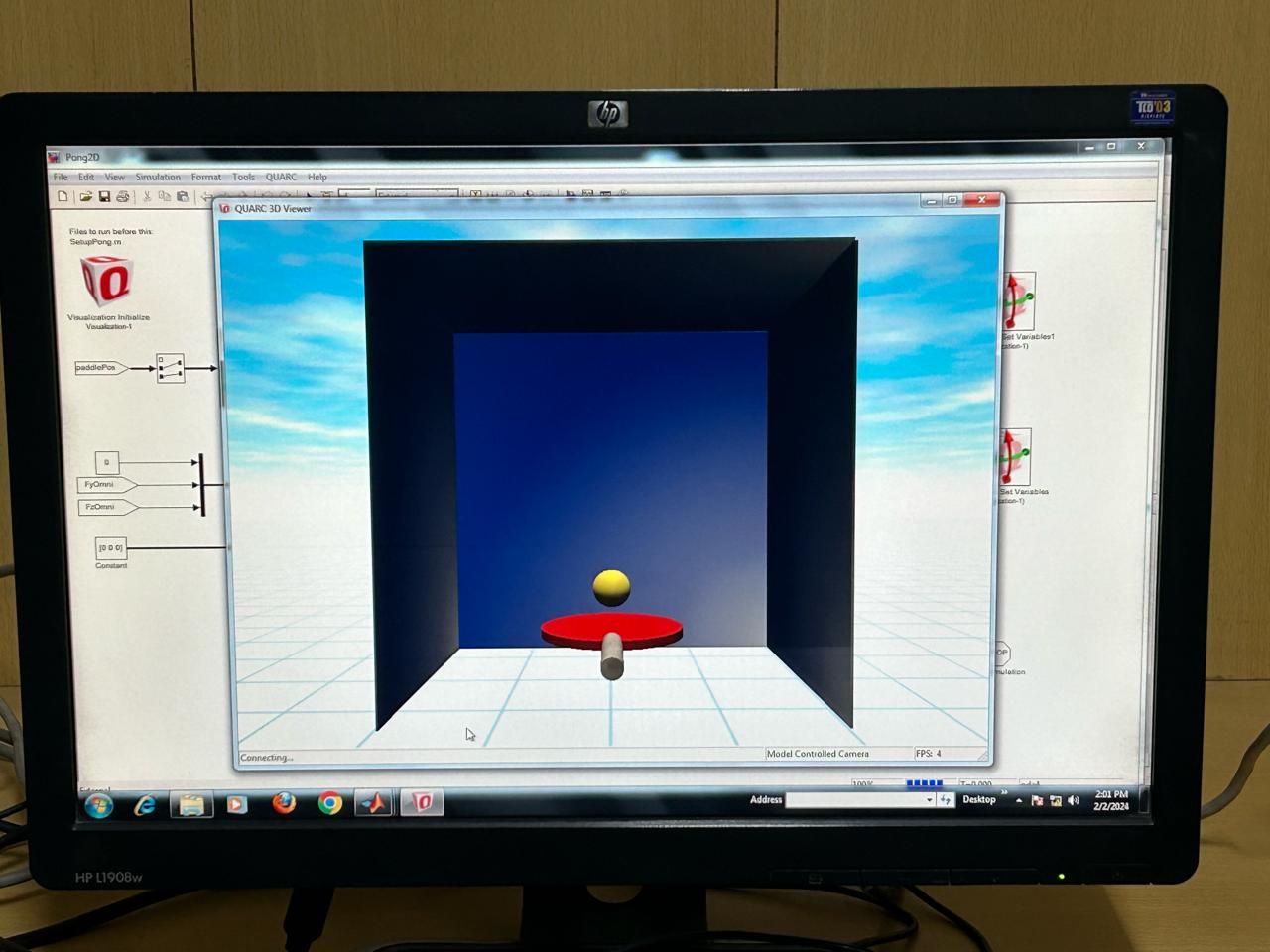
|  |
| --- |
| **Objective(s):**  TO ACCQUIRE DATA FROM AN HAPTIC FEEDBACK USING OMNI BUNDLE USING MATLAB SIMULINK. |

**Student Work Area**









**Experiment No: 4**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section**: 6th Semester – AIML-B (A3) |
| **Link to Code:** <https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/tree/main/NCU-CSL349%20-%20AI%20For%20Robotics%20-%20Lab%20Manual> |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective(s):**  TO PERFORM 2D/3D TRANSFORMATIONS AND VISUALIZE THEM USING MATLAB. |

**Student Work Area**

**Code:**

>> % Close all existing figures

close all;

% Define the vertices of the original square

originalSquare = [0 0; 1 0; 1 1; 0 1; 0 0];

% Define the rotation angle in radians (45 degrees)

theta = pi / 4;

% Define the 2D rotation matrix

rotationMatrix = [cos(theta) -sin(theta);

sin(theta) cos(theta)];

% Apply the rotation matrix to the vertices of the original square

% Note: Transpose originalSquare to match dimensions for matrix multiplication

rotatedSquare = (rotationMatrix \* originalSquare')';

% Begin plotting

figure; % Opens a new figure window

hold on; % Holds the plot for multiple plot commands

grid on; % Enables grid for better visualization

% Plot the original square

plot(originalSquare(:,1), originalSquare(:,2), 'b-', 'LineWidth', 2);

% Plot the rotated square

plot(rotatedSquare(:,1), rotatedSquare(:,2), 'r--', 'LineWidth', 2);

% Configure the plot

axis equal; % Ensure equal aspect ratio for both axes

xlabel('X axis');

ylabel('Y axis');

title('2D Rotation of a Square');

legend('Original', 'Rotated', 'Location', 'best');

hold off; % Release the plot hold

>>

**Output:**

A computer screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Experiment No: 5**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section**: 6th Semester – AIML-B (A3) |
| **Link to Code:** <https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/tree/main/NCU-CSL349%20-%20AI%20For%20Robotics%20-%20Lab%20Manual> |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective(s):**  To get femininized with basic of ARDIUNO and it type of implements basic led circuit |

**Student Work Area**

**Introduction to Arduino**

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's designed for hobbyists, tinkerers, and anyone interested in creating interactive projects. The term "Arduino" refers to the board itself, the company producing these boards, and the community that revolves around them. Arduino boards read inputs—such as light on a sensor, a finger on a button, or a Twitter message—and turn them into outputs like activating a motor, turning on an LED, or publishing something online.

**Common Types of Arduino Boards**

1. **Arduino Uno**: Features the ATmega328P microcontroller, 14 digital I/O pins, 6 analog inputs, and a 16 MHz ceramic resonator. It's a versatile board that's perfect for beginners.
2. **Arduino Mega**: Based on the ATmega2560, it provides 54 digital I/O pins, 16 analog inputs, and supports multiple serial communications, making it suitable for larger projects.
3. **LilyPad Arduino**: Utilizes the ATmega32u4, offers 9 digital I/O pins and is designed for wearables and e-textiles.
4. **Arduino BT (Bluetooth)**: Equipped with an ATmega328 and a Bluetooth module, ideal for wireless control projects.
5. **Arduino Nano**: A compact board based on the ATmega328, convenient for projects on a breadboard.
6. **Arduino Mini**: A smaller version of the Uno with similar functionality, designed for semi-permanent installation in objects or exhibitions.

**Key Components of an Arduino Board**

* **Digital I/O Pins**: Use these for input (like buttons) or output (like LEDs). Pins marked with **~** can perform PWM (Pulse Width Modulation) to simulate analog output.
* **Power LED Indicator**: Shows that the board is powered on.
* **Reset Button**: Resets the board and starts your program from the beginning.
* **Microcontroller**: The main IC that acts as the brain of the Arduino.
* **USB Connection**: Powers the board or connects it to your computer for programming.
* **Power Jack (Barrel Jack)**: Connects to an external power source like an AC-to-DC adapter.

**Programming Arduino**

Arduino sketches (programs) are written in the Arduino IDE, which simplifies coding in C/C++. After writing, the sketch is uploaded to the board for execution.

**Example Project: Blinking an LED**

This simple project uses the built-in LED to blink on and off.

**Code:**

// Setup the built-in LED pin as an output.

*void* setup()

{

    pinMode(LED\_BUILTIN, OUTPUT);

}

// The loop function runs continuously.

*void* loop()

{

    digitalWrite(LED\_BUILTIN, HIGH); // Turn on the LED

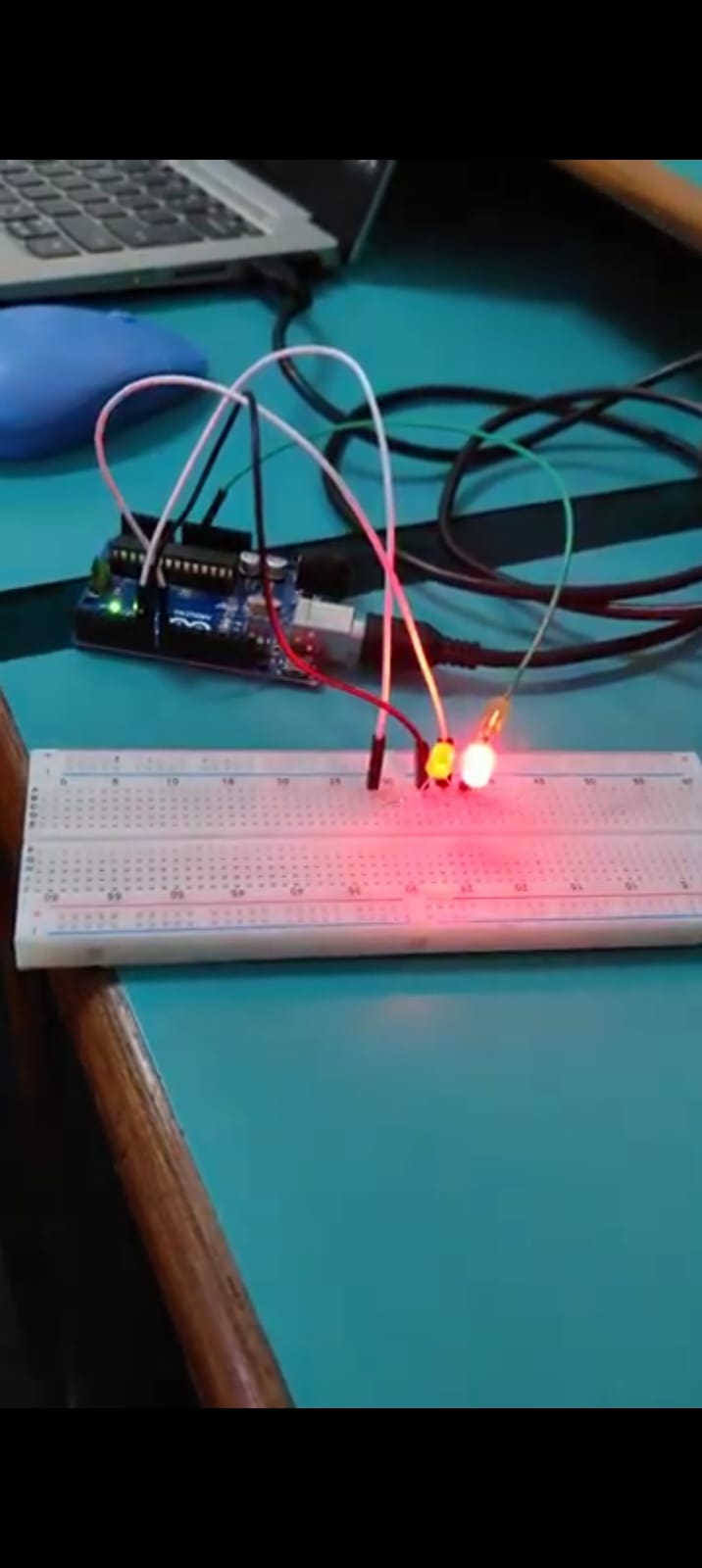
    delay(1000);                     // Wait for one second

    digitalWrite(LED\_BUILTIN, LOW);  // Turn off the LED

    delay(1000);                     // Wait for another second

}

**Setup & Output:**



**Experiment No: 6**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section**: 6th Semester – AIML-B (A3) |
| **Link to Code:** <https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/tree/main/NCU-CSL349%20-%20AI%20For%20Robotics%20-%20Lab%20Manual> |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective(s):**  To build a CRKT for traffic light system using Arduino based using program with and into IDE |

**Student Work Area**

**Components Needed:**

1. **Arduino Uno Board:** The main controller for the traffic light system.
2. **LEDs:** One each of red, yellow, and green.
3. **Resistors:** 220 ohms for each LED to limit the current and prevent damage.
4. **Breadboard:** For prototyping without soldering.
5. **Jumper Wires:** To connect the components on the breadboard.

**Code:**

// Define pin numbers for each LED

const *int* RED\_LED\_PIN = 9;

const *int* YELLOW\_LED\_PIN = 10;

const *int* GREEN\_LED\_PIN = 11;

// Define the duration of each light (in milliseconds)

const *int* GREEN\_LIGHT\_DURATION = 5000;  // Green light for 5 seconds

const *int* YELLOW\_LIGHT\_DURATION = 2000; // Yellow light for 2 seconds

const *int* RED\_LIGHT\_DURATION = 5000;    // Red light for 5 seconds

*void* setup()

{

    // Set LED pins as outputs

    pinMode(RED\_LED\_PIN, OUTPUT);

    pinMode(YELLOW\_LED\_PIN, OUTPUT);

    pinMode(GREEN\_LED\_PIN, OUTPUT);

}

*void* loop()

{

    // Sequence starts with green light

    digitalWrite(GREEN\_LED\_PIN, HIGH);

    delay(GREEN\_LIGHT\_DURATION);

    digitalWrite(GREEN\_LED\_PIN, LOW);

    // Then yellow light

    digitalWrite(YELLOW\_LED\_PIN, HIGH);

    delay(YELLOW\_LIGHT\_DURATION);

    digitalWrite(YELLOW\_LED\_PIN, LOW);

    // Followed by red light

    digitalWrite(RED\_LED\_PIN, HIGH);

    delay(RED\_LIGHT\_DURATION);

    // Yellow comes on briefly before green to signal readiness

    digitalWrite(RED\_LED\_PIN, LOW);

    digitalWrite(YELLOW\_LED\_PIN, HIGH);

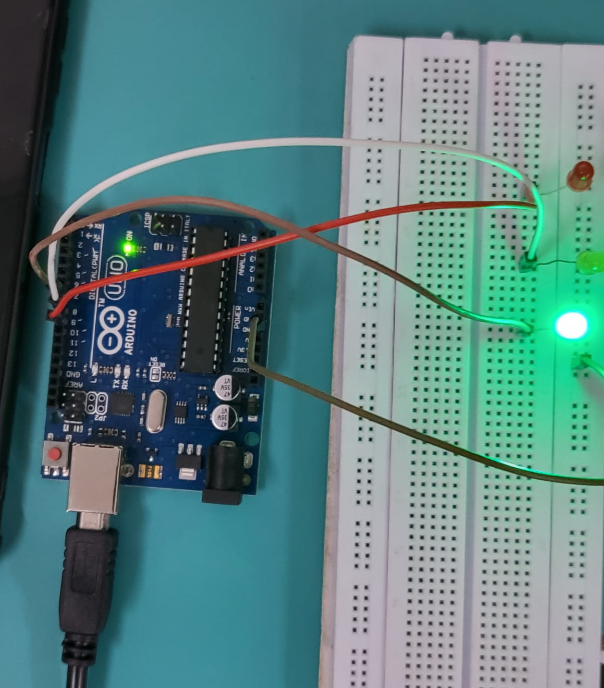
    delay(1000); // 1 second

    digitalWrite(YELLOW\_LED\_PIN, LOW);

    // All lights off before starting the next cycle

}

**Setup & Output:**

**Top of Form**

**Wiring Instructions:**

1. **Red LED:** Connect the longer leg (anode) to pin 9 via a 220-ohm resistor. Connect the shorter leg (cathode) to a GND (ground) pin on the Arduino.
2. **Yellow LED:** Connect the anode to pin 10 via a 220-ohm resistor, and the cathode to GND.
3. **Green LED:** Connect the anode to pin 11 via a 220-ohm resistor, and the cathode to GND.

**Uploading and Running the Code:**

1. Connect the Arduino to your computer using a USB cable.
2. Open the Arduino IDE, load the above code, and upload it to your Arduino board.
3. Once uploaded, the Arduino will start cycling through the lights, mimicking a traffic signal.

**Experiment No: 7**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section**: 6th Semester – AIML-B (A3) |
| **Link to Code:** <https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/tree/main/NCU-CSL349%20-%20AI%20For%20Robotics%20-%20Lab%20Manual> |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective(s):** To implement a Bluetooth system using Arduino UNO and HC-50bluetooth Module |

**Student Work Area**

**Hardware Setup:**

**Connect the HC-05 Bluetooth module to the Arduino Uno as follows:**

* HC-05 RX pin to Arduino Uno TX pin (pin 1)
* HC-05 TX pin to Arduino Uno RX pin (pin 0)
* HC-05 VCC pin to Arduino Uno 5V pin
* HC-05 GND pin to Arduino Uno GND pin

**Software Setup:**

1. Install the Arduino IDE: Ensure the Arduino IDE is installed on your computer. If it is not installed, download and install it from the official Arduino website.
2. Include the Software Serial Library:
   * Open the Arduino IDE.
   * Navigate to Sketch > Include Library > Manage Libraries.
   * In the Library Manager, search for Software Serial, and install it if it's not already installed.
3. Write the Arduino Sketch: Prepare your Arduino sketch to facilitate communication between the HC-05 module and other devices using Bluetooth.

**Code:**

#include <SoftwareSerial.h>

// Define pins for SoftwareSerial

const *int* RX\_PIN = 0; // Set RX to pin 0

const *int* TX\_PIN = 1; // Set TX to pin 1

// Initialize SoftwareSerial instance

SoftwareSerial BTSerial(RX\_PIN, TX\_PIN);

*void* setup()

{

    // Start communication with the computer

    Serial.begin(9600);

    // Start communication with the Bluetooth module

    BTSerial.begin(9600);

    // Print a startup message to the serial monitor

    Serial.println("Bluetooth Serial Communication Started");

}

*void* loop()

{

    // Check if there is data waiting in the serial monitor

    if (Serial.available())

    {

*char* data = Serial.read(); // Read the incoming byte

        BTSerial.write(data);      // Send that byte via Bluetooth

        Serial.print("Sent: ");    // Optional: Echo that byte back to the serial monitor

        Serial.println(data);

    }

    // Check if there is data received from the Bluetooth module

    if (BTSerial.available())

    {

*char* data = BTSerial.read(); // Read the incoming byte from Bluetooth

        Serial.write(data);          // Send that byte to the serial monitor

        Serial.print("Received: ");  // Optional: Indicate that a byte was received

        Serial.println(data);

    }

}

**Output:**

A screenshot of a computer program

Description automatically generated

**Setup:**

A blue circuit board with wires and a red light

Description automatically generated

**Experiment No: 8**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section**: 6th Semester – AIML-B (A3) |
| **Link to Code:** <https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/tree/main/NCU-CSL349%20-%20AI%20For%20Robotics%20-%20Lab%20Manual> |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective(s):** To implement of Roboanalyzer for dof 1,2,3 |

**Student Work Area**