DRONACHARYA
College of Engineering

LABORATORY MANUAL

B.Tech. Semester- VI

ROBOTICS ENGINEERING AND APPLICATION LAB
Subject code: LC-RA-312G

Prepared by:	**Checked by:**	**Approved by:**
Dr. R. Dheivanai	Mrs. Dimple Saproo	Name : Prof. (Dr.) Isha Malhotra
Sign.: ……………………..	**Sign.: ………………….**	**Sign.: ………………….**

DEPARTMENT OF ROBOTICS AND AUTOMATION
DRONACHARYA COLLEGE OF ENGINEERING
KHENTAWAS, FARRUKH NAGAR, GURUGRAM (HARYANA)

Table of Contents

Vision and Mission of the Institute

Vision:

To impart Quality Education, to give an enviable growth to seekers of learning, to groom them as World Class Engineers and Managers competent to match the expanding expectations of the Corporate World has been our ever enlarging vision extending to new horizons since the inception of Dronacharya College of Engineering.

Mission:

M1: To serve the society and improve the mode of life by imparting high quality education in the field of Engineering and Management catering to the explicit and implicit needs of the students, society, humanity and industry.

M2: To create an inspiring ambience that raises the motivation level for conducting quality research

M3: To provide an environment for acquiring ethical values and positive attitude.

Vision and Mission of the Department

Vision:

To be a globally recognized leader in robotics and automation education, research, and innovation, empowering students to excel in a technologically advanced world.

Mission:

M1: To provide high quality education and training in robotics and automation, equipping students with the knowledge, skills, and attitudes necessary for successful careers in the field.

M2: To foster a culture of innovation and entrepreneurship, encouraging student and faculty to develop and apply cutting-edge technologies in robotics and automation

M3: To conduct impactful research and development activities, addressing real-world challenges and advancing the field of robotics and automation

M4: To promote ethical practices, environmental sustainability and social responsibility in the deployment of technologies

M5. To collaborate with industry, academia and research organizations to create opportunities for industry-driven projects, internships, and placements ensuring the relevance of our programs and enhancing industry readiness or our graduates

Programme Educational Objectives (PEOs)

PEO1: To provide innovative and state-of-the-art solutions solve complex problems in automation, robotics and allied fields, and design high quality systems for diverse applications.

PEO2: To develop research oriented analytical ability among students and to prepare them for making technical contribution to the society.

PEO3: To continue life-long learning and pursue professional development opportunities like graduate degrees or professional studies to adapt to the evolving technological changes..

PEO4: To prepare the students work in diverse, multi-disciplinary teams and possess leadership skills, ethical standards, environmental concerns and social awareness.

PEO5: To Re-learn and innovate in ever-changing global economic and technological environments on the 21st century

Programme Outcomes (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and software tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Programme Specific Outcomes (PSOs)

PSO1: Identify the needs, analyze, design and develop simple robotic systems and programs for diverse applications in real time.

PSO2: Design, select and integrate appropriate automation and robotic subsystems for multi-domain engineering and integrate software applications tools.

PSO3: Develop impactful engineering solutions by using research-based knowledge and research methods in the fields of advanced robotics and other relevant fields.

PSO4: Evaluate existing engineering elements and processes, identifying areas for improvement. Propose innovative robotic and automation solutions to enhance the performance and efficiency of conventional systems.

PSO5: Identify suitable sensing, interfacing, control, actuation, and communication technologies to integrate various subsystems. Develop robots capable of analyzing data and implementing automated solutions through seamless connectivity between different components.

University Syllabus

1. Determination of maximum and minimum position of links.
2. Verification of transformation (Position and orientation) with respect to gripper and world coordinate system
3. Estimation of accuracy, repeatability and resolution.
4. Robot programming and simulation for pick and place
5. Robot programming and simulation for colour identification
6. Robot programming and simulation for Shape identification
7. Robot programming and simulation for machining (cutting, welding)
8. Robot programming and simulation for writing practice
9. Robot programming and simulation for any industrial process (Packaging, Assembly)
10. Robot programming and simulation for multi process.

Course Outcomes (COs)

Upon successful completion of the course, the student will be able to:

CO1: - Use of any robotic simulation software to model the different types of robots and calculate work volume for different robots

CO-PO Mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	1	2	1	2				3	2	2	1	1
Average	**1**	**2**	**1**	**2**				**3**	**2**	**2**	**1**	**1**

CO-PSO Mapping

	PSO1	PSO2	PSO3
CO1	2	2	1
Average	**2**	**2**	**1**

Course Overview

This is the fundamental course for the Robotics and Automation program. The aim of this course is to give an overview of basic tools and techniques used for Industrial applications. IT emphasizes the physical understanding of robot kinematics and dynamics, differential motion, design and control of robotic arms.

This course supports the students to design and develop the systems, automation and robotics for various industrial applications such as identifying the shape, colour of an object.

Simulation software like Tinkercad helps to do the simulation of any industrial process like packaging, assembly, pick and place the objects from one place to another place, etc.

From this subject, students can make use of any robotic simulation software to model the different types of robots and they can able to calculate the work volume for different robots.

List of Experiments mapped with COs

S. No.	Name of the Experiments	Course Outcome
1	Determination of maximum and minimum position of links	CO1
2	Verification of transformation (Position and orientation) with respect to gripper and world coordinate system	CO1
3	Estimation of accuracy, repeatability and resolution	CO1
4	Robot programming and simulation for pick and place	CO1
5	Robot programming and simulation for Color identification	CO1
6	Robot programming and simulation for Shape identification	CO1
7	Robot programming and simulation for machining (cutting, welding)	CO1
8	Robot programming and simulation for writing practice	CO1
9	Robot programming and simulation for any industrial process (Packaging, Assembly)	CO1
10	Robot programming and simulation for multi process.	CO1

Dos and DONT's

Dos

1. Login-on with your username and password.
2. Log off the computer every time when you leave the Lab.
3. Arrange your chair properly when you are leaving the lab.
4. Put your bags in the designated area.

DON'Ts

1. Do not share your username and password.
2. Do not remove or disconnect cables or hardware parts.
3. Do not personalize the computer setting.
4. Do not run programs that continue to execute after you log off.
5. Do not download or install any programs, games or music on computer in Lab.
6. Personal Internet use chat room for Instant Messaging (IM) and Sites is strictly prohibited.
7. No Internet gaming activities allowed.
8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab

General Safety Precautions

Precautions (In case of Injury or Electric Shock)

1. To break the victim with live electric source, use an insulator such as fire wood or plastic to break the contact. Do not touch the victim with bare hands to avoid the risk of electrifying yourself.

2. Unplug the risk of faulty equipment. If main circuit breaker is accessible, turn the circuit off.

3. If the victim is unconscious, start resuscitation immediately, use your hands to press the chest in and out to continue breathing function. Use mouth-to-mouth resuscitation if necessary.

4. Immediately call medical emergency and security. Remember! Time is critical; be best.

Precautions (In case of Fire)

1. Turn the equipment off. If power switch is not immediately accessible, take plug off.

2. If fire continues, try to curb the fire, if possible, by using the fire extinguisher or by covering it with a heavy cloth if possible isolate the burning equipment from the other surrounding equipment.

3. Sound the fire alarm by activating the nearest alarm switch located in the hallway.

4. Call security and emergency department immediately:

Emergency: Reception

Security : Main Gate

Guidelines to students for report preparation

All students are required to maintain a record of the experiments conducted by them. Guidelines for its preparation are as follows: -

1) All files must contain a title page followed by an index page. **The files will not be signed by the faculty without an entry in the index page.**

2) Student's Name, Roll number and date of conduction of experiment must be written on all pages.
3) For each experiment, the record must contain the following

(i) Aim/Objective of the experiment

(ii) Apparatus required with specification and Name plate details

(iii) Circuit diagrams, procedures, observations and calculations

(v) Results/ output

Note:

1. Students must bring their lab record along with them whenever they come for the lab.

2. Students must ensure that their lab record is regularly evaluated.

Lab Assessment Criteria

An estimated 10 lab classes are conducted in a semester for each lab course. These lab classes are assessed continuously. Each lab experiment is evaluated based on 5 assessment criteria as shown in following table. Assessed performance in each experiment is used to compute CO attainment as well as internal marks in the lab course.

Grading Criteria	Exemplary (4)	Competent (3)	Needs Improvement (2)	Poor (1)
AC1: **Pre-Lab written work (this may be assessed through viva)**	Complete procedure with underlined concept is properly written	Underlined concept is written but procedure is incomplete	Not able to write concept and procedure	Underlined concept is not clearly understood
AC2: **Program Writing/ Modelling**	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied, Program/solution written is readable	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied	Assigned problem is properly analyzed & correct solution designed	Assigned problem is properly analyzed
AC3: Identification & Removal of errors/ bugs	Able to identify errors/ bugs and remove them	Able to identify errors/ bugs and remove them with little bit of guidance	Is dependent totally on someone for identification of errors/ bugs and their removal.	Unable to understand the reason for errors/ bugs even after they are explicitly pointed out
AC4: **Execution & Demonstration**	All variants of input /output are tested, Solution is well demonstrated and implemented concept is clearly explained	All variants of input /output are not tested, However, solution is well demonstrated and implemented concept is clearly explained	Only few variants of input /output are tested, Solution is well demonstrated but implemented concept is not clearly explained	Solution is not well demonstrated and implemented concept is not clearly explained
AC5: **Lab Record Assessment**	All assigned problems are well recorded with objective, design constructs and solution along with Performance analysis using all variants of input and output	More than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	Less than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	Less than 40 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output

LAB EXPERIMENTS

LAB EXPERIMENT 1

Determination of maximum and minimum position of link

AIM: To determine the maximum and minimum position of links

Materials Required:

- A robotic arm or a similar mechanical system with interconnected links
- Position sensors (Ex. Encoders, potentiometers) to measure the position of each link
- A computer or data acquisition to measure the physical position of the links
- A ruler or tape measure to measure the physical position of the links
- A set of weights or other objects to load the system and test its limits

Pre-Experiment Questions

1. What is a link in a mechanism
2. Why is it important to determine the maximum and minimum positions of links in a mechanism?

Procedure:

1. Install the position sensors on each link of the robotic arm, following the manufacturer's instructions or previous lab notes. Make sure that the sensors are calibrated and provide accurate readings.
2. Connect the position sensors to the computer or data acquisition system and configure the software to collect and store the position data.
3. Identify the maximum and minimum range of motion of each link, based on the physical constraints of the system and the specifications of the sensors. For example, if the robotic arm has six links and each link can rotate up to 180 degrees, the maximum range of motion of each link would be from -90 to +90 degrees.
4. Start with one link and move it slowly from the minimum position to the maximum position, while recording the position data. Repeat the process a few times to ensure consistency and repeatability of the measurements.
5. Plot the position data on a graph, with the position on the y-axis and time or angle on the x-axis. Identify any trends or patterns in the data, such as non-linearities, oscillations, or saturation points.

6. Repeat the process for all the links of the robotic arm, one by one or in parallel, depending on the available equipment and resources. Make sure to record the data and analyse it using appropriate statistical or mathematical tools.
7. Load the system with additional weights or other objects to test its limits and see how it behaves under different loads. Repeat the measurements and analysis as before, and compare the results to the unloaded case.
8. Evaluate the accuracy and precision of the measurements and the limitations of the experimental setup. Consider possible sources of error or uncertainty, such as sensor noise, measurement drift, or mechanical hysteresis, and try to minimize or correct for them.

Formula:

Calculate the maximum and minimum positions of each link relative to the reference link using the following formulae:

- Maximum Position = length pf link x sin (angle between links)
- Minimum Position = length pf link x sin (180-angle between links)

Results:

The laboratory experiment should provide a quantitative and qualitative assessment of the maximum and minimum position of the links of the mechanical system under study. The position data can be used to calculate the range of motion, the velocity and acceleration profiles, and other relevant parameters of the system. The loaded tests can also reveal the dynamic behaviour and the robustness of the system under various operating conditions. The results can be compared to the theoretical models or simulations of the system, if available, or used to improve the design and performance of the system in practice.

Post-Experiment Questions:

1. What are the materials required to determine the maximum and minimum positions of links in a mechanism
2. How do you calculate the maximum and minimum positions of a link
3. What is the purpose of creating a graph of the maximum and minimum positions of each lin

LAB EXPERIMENT 2

Verification of transformation (position and orientation) with respect to gripper and world coordinate system

Aim

To verify the transformation (position and orientation) with respect to gripper and world coordinate system

Equipment Required:

- Robot arm with a gripper
- Markers
- Camera System
- Computer with control software and computer vision software

Pre-experiment Questions

1. What is the purpose of this experiment?
2. What are the potential applications of this experiment in robotics?
3. What equipment is required for this experiment?

Procedure:

1. Set up the robot arm in the laboratory, with the gripper attached to the end effector.
2. Place the markers at known positions in the laboratory workspace. These markers should be visible to the robot's camera system.
3. Use the robot's control software to move the gripper to various positions and orientations in the workspace, recording the position and orientation of the gripper for each movement.
4. Use the robot's camera system to capture images of the markers in the workspace, and use computer vision techniques to calculate the position and orientation of the markers in the gripper's coordinate system.

5. Use the recorded gripper positions and orientations, along with the marker positions and orientations in the gripper's coordinate system, to calculate the transformation matrix between the gripper and the world coordinate systems.
6. Verify the accuracy of the transformation matrix by comparing the calculated positions and orientations of the markers in the world coordinate system to their known positions.
7. Repeat the experiment for different gripper positions and orientations to test the robustness of the transformation matrix.
8. Finally, use the verified transformation matrix to program the robot arm to perform various tasks, such as picking up and moving objects in the workspace

RESULT:

Thus we verified the transformation (position and orientation) with respect to gripper and world coordinate system

Post- Experiment Questions

1. What are markers and why are they used in this experiment?
2. How do you calculate the transformation matrix between the gripper and the world coordinate systems?
3. How do you verify the accuracy of the transformation matrix?
4. What is the significance of testing the robustness of the transformation matrix?
5. How can the verified transformation matrix be used in programming the robot arm?

LAB EXPERIMENT 3

Estimation of accuracy, repeatability and resolution

AIM:

To determine estimation of accuracy, repeatability and resolution

Equipment required:

- Measurement instrument (e.g. ruler, caliper, scale)
- Representative samples of the measurement variable
- Spreadsheet or data analysis tool

Pre-Experiment Questions

1. What is the purpose of this experiment?
2. What are measurement variables and why are they important in this experiment?
3. What equipment is required for this experiment?

Procedure:

- Set up the experimental apparatus in the laboratory.
- Define the measurement variables that will be used to estimate accuracy, repeatability, and resolution.
- Select a representative set of measurement points that span the range of values of the variables.
- Make measurements at each of the selected measurement points, using a precise measurement instrument.
- Repeat the measurements at each point multiple times to estimate repeatability.
- Compare the measured values to the true values, if known, or to a reference standard to estimate accuracy.
- Determine the resolution by measuring the smallest change in the variable that can be detected by the measurement instrument.
- Analyze the measurement data to calculate the accuracy, repeatability, and resolution.

Theory:

- **Experimental Apparatus**: The experimental apparatus should be designed to be as precise and accurate as possible for the given measurement variables. For example, if measuring the length of an object, a high-precision ruler or caliper should be used.
- **Measurement Variables**: The measurement variables should be carefully selected to

represent the critical aspects of the experiment. For example, in a material strength test, the measurement variables may include tensile strength, yield strength, and fracture toughness.

- **Measurement Points:** Select measurement points that span the range of values of the variables to get a representative estimate of accuracy, repeatability, and resolution.
- **Measurement:** Make measurements at each of the selected measurement points, using a precise measurement instrument. Record the measured values in a spread sheet or data analysis tool.
- **Repeatability**: Repeat the measurements at each point multiple times to estimate repeatability. Calculate the mean and standard deviation of each set of measurements.
- **Accuracy:** Compare the measured values to the true values, if known, or to a reference standard to estimate accuracy. Calculate the difference between the measured value and the true/reference value.
- **Resolution**: Determine the resolution by measuring the smallest change in the variable that can be detected by the measurement instrument. This can be done by gradually increasing or decreasing the value of the variable until a change is no longer detected.
- **Data Analysis:** Analyze the measurement data to calculate the accuracy, repeatability, and resolution. Calculate the mean and standard deviation of the measurements, and compare them to the true/reference values. Calculate the smallest detectable change in the variable to estimate resolution.

RESULT:

Thus we determined the estimation of accuracy, repeatability and resolution

Post- Experiment Questions

1. How do you select the representative set of measurement samples?
2. What is repeatability and how is it estimated in this experiment?
3. What is accuracy and how is it estimated in this experiment?
4. What is resolution and how is it determined in this experiment?
5. How do you analyze the measurement data to calculate the accuracy, repeatability, and resolution?
6. What are the potential sources of error in this experiment?
7. How can the results of this experiment be used to improve the accuracy and precision of future measurements in the laboratory
8.

LAB EXPERIMENT 4

Robot Programming and simulation for pick and place

AIM:

To do the Robot Programming and simulation for pick and place

Materials Required:

- Arduino UNO or equivalent
- Servo motors (2 or more)
- Ultrasonic sensor
- Breadboard and jumper wires
- USB cable
- Computer with Arduino IDE software installed

Pre-Experiment Questions

1. What is the purpose of the Servo library in this code?
2. What is the purpose of the buttonPin variable in this code?
3. What is the purpose of the motorPin variable in this code?

Procedure:

Step 1: Build the robot arm

- Build the robot arm using servo motors and other materials.
- Connect the servo motors to the Arduino UNO board and make sure they are properly mounted.

Step 2: Connect the ultrasonic sensor

- Connect the ultrasonic sensor to the breadboard and Arduino board.
- Connect the power, ground, and signal pins of the sensor to the appropriate pins on the board.

Step 3: Program the Arduino

- Open the Arduino IDE software on your computer.

- Write a code to control the robot arm and ultrasonic sensor.
- The code should instruct the robot arm to move in a certain direction when the ultrasonic sensor detects an object at a certain distance.
- Use the Servo library to control the servo motors and the NewPing library to read the ultrasonic sensor.
- Verify and upload the code to the Arduino board.

Step 4: Test the robot arm

- Power up the Arduino board and run the program.
- Test the robot arm by placing objects within the range of the ultrasonic sensor and observing how it responds.
- You can adjust the code to make the robot arm move in a certain way depending on the position of the object.

Step 5: Simulate the robot arm

- To simulate the robot arm, you can use a software tool like Tinkercad or SolidWorks.
- Create a virtual model of the robot arm and connect it to a virtual Arduino board.
- Write a code to control the virtual robot arm and simulate its movement based on the inputs from the virtual ultrasonic sensor.
- Test and refine the code until the virtual robot arm is able to perform the desired tasks.

Code using Arduino:

```
#include <Servo.h>
Servo servoX;
Servo servoY;
const int buttonPin = 2;
const int servoXPin = 9;
const int servoYPin = 10;
const int motorPin = 3;
int buttonState = 0;
void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(motorPin, OUTPUT);
  servoX.attach(servoXPin);
  servoY.attach(servoYPin);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    // move to pick up position
    servoX.write(90);
    servoY.write(45);
    delay(1000);
    // activate motor to pick up object
    digitalWrite(motorPin, HIGH);
    delay(500);
    digitalWrite(motorPin, LOW);
    delay(500);
    // move to drop off position
    servoX.write(0);
    servoY.write(90);
    delay(1000);
    // activate motor to drop off object
    digitalWrite(motorPin, HIGH);
    delay(500);
    digitalWrite(motorPin, LOW);
    delay(500);
  }
}
```

RESULT:

Thus we completed the Robot Programming and simulation for pick and place

Post-Experiment Questions

1. What is the purpose of the delay function in this code? Can it be replaced with other functions?
2. How would you modify this code to pick up and drop off objects at different position

LAB EXPERIMENT 5

Robot Programming and simulation for colour identification

AIM: To do the robot programming and simulation for colour identification

Connecting the Hardware:

1. Install the Arduino Nano at Breadboard
2. Connect the Nano 5V output and GND at both Power Rails
3. Connect the TSC3200 Sensor as bellow:
   - S0 ==> Nano pin D4
   - S1 ==> Nano pin D5
   - S2 ==> Nano pin D6
   - S3 ==> Nano pin D7
   - OUT ==> Nano Pin D8
   - EN ==> GND
   - VCC ==> +5V
   - GND ==> GND
4. Connect the I2C LCD 2/16 Serial Display:
   - SDA ==> Nano Pin A4
   - SCL ==> Nano Pin A5

Pre-Experiment Questions

1. What is an RGB color sensor and how does it work
2. How does the Adafruit_RGBLCDShield library work in Arduino
3. Can you explain how the program controls the LED based on the color detected by the sensor

Arduino Code:

The first thing to define is the frequency scaling as defined at the table showed above. Pins S0 and S1 are used for that. Scaling the output frequency is useful to optimize the sensor readings for various frequency counters or microcontrollers. We will set S0 and S1, both in HIGH (100%):

```
digitalWrite(s0,HIGH);

digitalWrite(s1,HIGH);
```

Next thing to do is to select the color to be read by the photodiode (Red, Green, or Blue), we use the control pins S2 and S3 for that. As the photodiodes are connected in parallel, setting the S2 and S3

LOW and HIGH in different combinations allows you to select different photodiodes, as showed at above table.

```
digitalWrite(s2, LOW);

digitalWrite(s3, LOW);

red = pulseIn(outPin, LOW); // Reading RED component of color

digitalWrite(s2, HIGH);

digitalWrite(s3, HIGH);

grn = pulseIn(outPin, LOW); // Reading GREEN component of color

digitalWrite(s2, LOW);

digitalWrite(s3, HIGH);

blu = pulseIn(outPin, LOW); // Reading BLUE component of color
```

On the final code, we will read a few times each one of the RGB components and take an average, so we can reduce the error if one of the readings is bad.

Once we have the 3 components (RGB), we must define what color is that. The way to do it is to previously calibrate the project. You can use a known colored test paper or object and read the 3 components generated.

```
void getColor()

{

  readRGB();

     if (red > 8 && red < 18  && grn > 9 && grn < 19   && blu > 8 && blu < 16)  color =
"WHITE";

  else if (red > 80 && red < 125 && grn > 90 && grn < 125  && blu > 80 && blu < 125)
color = "BLACK";

  else if (red > 12 && red < 30  && grn > 40 && grn < 70   && blu > 33 && blu < 70)  color
= "RED";

  else if (red > 50 && red < 95  && grn > 35 && grn < 70   && blu > 45 && blu < 85)  color
= "GREEN";
```

```
  else if (red > 10 && red < 20   &&  grn > 10 && grn < 25    &&  blu > 20 && blu < 38)   color
= "YELLOW";

  else if (red > 65 && red < 125  &&  grn > 65 && grn < 115   &&  blu > 32 && blu < 65)
color = "BLUE";

  else  color = "NO_COLOR";

}
```

Here 6 colours are predefined: White, Black, Red, Green, Yellow, and Blue. As the ambient light goes down, the parameters tend to go higher.

Inside the loop(), it is defined to display the readings at LCD each 1 second.

https://mjrobot.org/arduino-color-detection/

In simple we can do this method using Arduino and the Tinkercad simulation tool

- First, set up the circuit on Tinkercad by placing an RGB color sensor and an LED on the breadboard, connecting them to the Arduino as shown in the following diagram
- Next, open the Arduino IDE and upload the following code to the Arduino board

Arduino Code

```
// Libraries
#include <Wire.h>
#include <Adafruit_RGBLCDShield.h>
#include <Adafruit_ColorSensor.h>
// Initialize the LCD
Adafruit_RGBLCDShield lcd = Adafruit_RGBLCDShield();
// Initialize the color sensor
Adafruit_ColorSensor colorSensor = Adafruit_ColorSensor();
void setup() {
  // Initialize the LCD
  lcd.begin(16, 2);
  lcd.setBacklight(255, 255, 255);
  // Initialize the color sensor
  colorSensor.begin();
}
void loop() {
  // Read the color from the sensor
  uint16_t red, green, blue;
  colorSensor.getColor(&red, &green, &blue);
  // Convert the color to a string and display it on the LCD
  char colorString[16];
  sprintf(colorString, "#%02x%02x%02x", red / 256, green / 256, blue / 256);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Color: ");
  lcd.print(colorString);
  // Control the LED based on the color
  if (red > blue && red > green) {
    // Red
    analogWrite(3, 255);
    analogWrite(5, 0);
    analogWrite(6, 0);
  } else if (green > red && green > blue) {
    // Green
    analogWrite(3, 0);
    analogWrite(5, 255);
    analogWrite(6, 0);
  } else {
    // Blue
    analogWrite(3, 0);
    analogWrite(5, 0);
    analogWrite(6, 255);
  }
  // Delay for a short time
  delay(100);
}
```

RESULT:

Hence we have done the robot programming and simulation for colour identification

Post-Experiment Questions:

1. How would you modify the code to detect and display the intensity of each color (red, green, and blue) separately?
2. How would you modify the circuit to add additional LEDs that light up for different colors?

LAB EXPERIMENT 6

Robot Programming and simulation for shape identification

AIM: To do the robot programming and simulation for shape identification

1. First, set up the circuit on Tinkercad by placing an ultrasonic sensor and three LEDs on the breadboard, connecting them to the Arduino as shown in the following diagram:
2. Next, open the Arduino IDE and upload the following code to the Arduino board

Pre-Experiment Questions

1. What is an ultrasonic sensor and how does it work?
2. How does the Servo library work in Arduino?

Arduino code:

```
// Libraries

#include <Servo.h>

// Initialize the servo

Servo servo;

// Initialize the LED pins

const int LED_1_PIN = 2;

const int LED_2_PIN = 3;

const int LED_3_PIN = 4;

// Initialize the ultrasonic sensor pins

const int TRIGGER_PIN = 5;

const int ECHO_PIN = 6;

void setup() {

  // Initialize the servo

  servo.attach(9);

  // Initialize the LED pins
  pinMode(LED_1_PIN, OUTPUT);

  pinMode(LED_2_PIN, OUTPUT);

  pinMode(LED_3_PIN, OUTPUT);

  // Initialize the ultrasonic sensor pins

  pinMode(TRIGGER_PIN, OUTPUT);

  pinMode(ECHO_PIN, INPUT);

  // Initialize the serial communication

  Serial.begin(9600);

}

void loop() {

  // Move the servo to scan the area

  for (int angle = 0; angle <= 180; angle += 10) {

    servo.write(angle);

    delay(100);

    // Check the distance to the nearest object
    long duration, distance;
    digitalWrite(TRIGGER_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIGGER_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIGGER_PIN, LOW);
    duration = pulseIn(ECHO_PIN, HIGH);
    distance = duration / 58.2;

    // Identify the shape based on the distance
    if (distance > 0 && distance <= 5) {
      // Circle
      digitalWrite(LED_1_PIN, HIGH);
      digitalWrite(LED_2_PIN, LOW);
      digitalWrite(LED_3_PIN, LOW);
      Serial.println("Circle detected.");
      delay(500);
    } else if (distance > 5 && distance <= 10) {
```

```
    // Square
    digitalWrite(LED_1_PIN, LOW);
    digitalWrite(LED_2_PIN, HIGH);
    digitalWrite(LED_3_PIN, LOW);
    Serial.println("Square detected.");
    delay(500);
  } else if (distance > 10 && distance <= 15) {
    // Triangle
    digitalWrite(LED_1_PIN, LOW);
    digitalWrite(LED_2_PIN, LOW);
    digitalWrite(LED_3_PIN, HIGH);
    Serial.println("Triangle detected.");
    delay(500);
  } else {
    // No shape detected
    digitalWrite(LED_1_PIN, LOW);
    digitalWrite(LED_2_PIN, LOW);
    digitalWrite(LED_3_PIN, LOW);
    Serial.println("No shape detected.");
    delay(500);
  }

 }

}
```

Finally, run the simulation on Tinkercad and test the program by placing different shaped objects in front of the ultrasonic sensor. The LEDs should light up to indicate the detected shape and the serial monitor should display the shape name.

```
  #include <Wire.h>

  #include <Adafruit_Sensor.h>

  #include <Adafruit_HCSR04.h>

  #include <Servo.h>

  Servo myservo; // create servo object to control a servo

  int pos = 0; // variable to store the servo position

  #define trigPin 13 // define the pins for the ultrasonic sensor

  #define echoPin 12

  Adafruit_HCSR04 us = Adafruit_HCSR04(trigPin, echoPin); // create object for ultrasonic sensor
```

```
int dist; // variable to store the distance measured by the sensor
void setup() {
  Serial.begin(9600); // initialize serial communication at 9600 baud
  myservo.attach(9); // attach the servo on pin 9 to the servo object
}
void loop() {
  pos = 90; // set the initial position of the servo to 90 degrees (facing forward)
  myservo.write(pos); // move the servo to the initial position
  delay(1000); // wait for the servo to reach the position
  dist = us.ping_cm(); // measure the distance using the ultrasonic sensor
  Serial.print("Distance: ");
  Serial.println(dist); // print the distance to the serial monitor
  if (dist < 10) { // if an object is detected within 10 cm
    pos = 0; // move the servo to the left (0 degrees)
    myservo.write(pos);
    delay(1000);
    dist = us.ping_cm(); // measure the distance again
    Serial.print("Distance: ");
    Serial.println(dist);
    if (dist < 10) { // if an object is still detected within 10 cm
      Serial.println("Square"); // identify the shape as a square

    } else {

      Serial.println("Triangle"); // identify the shape as a triangle
    }
  } else {
    pos = 180; // move the servo to the right (180 degrees)
    myservo.write(pos);
    delay(1000);
    dist = us.ping_cm(); // measure the distance again
    Serial.print("Distance: ");
    Serial.println(dist)
    if (dist < 10) { // if an object is detected within 10 cm
      Serial.println("Circle"); // identify the shape as a circle
    } else {
      Serial.println("Unknown shape"); // identify the shape as unknown
    }
  }
    }
```

This program uses an ultrasonic sensor to measure the distance between the sensor and an object in front of it. A servo motor is used to rotate the sensor to different positions. If an object is detected within 10 cm of the sensor, the servo moves to the left or right to get a better view of the object, and the distance is measured again. Based on the distance measurements, the program identifies the shape of the object as a square, triangle, circle, or unknown.

RESULT: Thus we have done the robot programming and simulation for shape identification

Post- Experiment Questions

1. Can you explain how the program identifies the shape based on the distance measured by the ultrasonic sensor?
2. How would you modify the code to detect additional shapes?
3. How would you modify the circuit to add a buzzer that sounds

LAB EXPERIMENT 7

Robot Programming and simulation for machining (cutting, welding)

AIM:

To do the Robot Programming and simulation for machining (cutting, welding)

Equipment Required

- Arduino UNO board
- Motor driver shield
- Stepper motors (two or more)
- End effector (cutting or welding tool)
- Computer with Arduino IDE installed
- Breadboard and jumper wires
- Power supply (for motors and Arduino)

Pre-Experiment Questions

1. What is the purpose of the Robot Programming and Simulation for Machining (Cutting, Welding) laboratory experiment using Arduino?
2. What components are required to build the robotic system?

Procedure

1. Design the robotic system: The first step is to design the robotic system that will be used for machining. The design should include the number and type of motors required, the end effector (cutting or welding tool), and any other necessary components.
2. Build the robotic system: After designing the robotic system, the next step is to build it. This involves assembling the components and wiring them up to the Arduino board.
3. Program the Arduino board: The next step is to program the Arduino board using the Arduino IDE. The program should include the control algorithm for the robotic system, which will be used to control the motors and the end effector. The program should also include any necessary sensor inputs and outputs.

4. Test the robotic system: After programming the Arduino board, the next step is to test the robotic system. This involves running the program and observing the behaviour of the robotic system. Any issues that arise during testing should be identified and addressed.
5. Perform machining tasks: Once the robotic system is working correctly, the next step is to perform machining tasks such as cutting and welding. The end effector should be programmed to move in a specific pattern, which will be used to cut or weld the work piece.

Theory:

1. Choose a robot platform: There are several robot platforms available for machining and welding. You can choose from industrial robots, hobby robots, or even build your own robot from scratch. Arduino is a popular platform for hobby robotics, and there are several kits available that can be used for this project.
2. Choose a programming language: There are several programming languages that can be used to program a robot, including C++, Python, and Java. Arduino uses a simplified version of C++, which makes it an ideal platform for beginners. You can also use graphical programming languages like Scratch or Blockly to program the robot.
3. Install the necessary software: You will need to install the Arduino IDE on your computer, which is a free software that allows you to write and upload code to the Arduino board. You will also need to install a simulation software like RoboDK, which allows you to simulate the robot movements and check for any errors before running the program on the actual robot.
4. Write the program: Once you have chosen your programming language and installed the necessary software, you can start writing the program. You will need to define the robot movements, such as the cutting or welding path, and the parameters such as speed, acceleration, and direction. You can also add sensors to detect any obstacles or errors during the process.
5. Test the program: Before running the program on the actual robot, you should test it on the simulation software to ensure that it is working correctly. Make any necessary adjustments to the program and repeat the testing until it is error-free.
6. Run the program on the robot: Once you are satisfied with the program, you can upload it to the Arduino board and run it on the actual robot. Make sure that the robot is calibrated correctly and all the safety precautions are in place before running the program.

7. Evaluate the results: After running the program, you should evaluate the results to see if it meets the desired outcome. You can also make any necessary adjustments to the program for future use.

Arduino Code:

```
#include <AccelStepper.h> // Import the AccelStepper library
// Define the motor pins
#define motorPin1 8
#define motorPin2 9
#define motorPin3 10
#define motorPin4 11

// Define the end effector pin
#define effectorPin 12
// Create the AccelStepper objects for the X and Y axis
AccelStepper x_axis(AccelStepper::FULL4WIRE, motorPin1, motorPin2, motorPin3, motorPin4);
AccelStepper y_axis(AccelStepper::FULL4WIRE, motorPin1, motorPin2, motorPin3, motorPin4);
// Define the end effector object
Servo effector;
// Define the position variables
int xPos = 0;
int yPos = 0;
void setup() {
  // Set the motor speeds and acceleration
  x_axis.setMaxSpeed(2000);
  x_axis.setAcceleration(1000);
  y_axis.setMaxSpeed(2000);
  y_axis.setAcceleration(1000);
  // Attach the end effector to the pin
  effector.attach(effectorPin);
}
```

```
void loop() {
  // Move the robot to the desired position
  x_axis.moveTo(xPos);
  y_axis.moveTo(yPos);

  // Check if the motors have reached their target position
  if (x_axis.distanceToGo() == 0 && y_axis.distanceToGo() == 0) {
    // Perform the machining task (in this case, turn on the end effector)
    effector.write(90);
  }
  // Increment the position variables (for a simple example)
  xPos += 100;
  yPos += 100;
  // Delay to allow the motors to move
  delay(1000);
}
```

This code uses the AccelStepper library to control the stepper motors for the X and Y axis. The end effector is controlled using the Servo library. The robot is moved to a desired position by setting the target position for each motor using the moveTo() function. Once the motors have reached their target position, the end effector is activated by setting its angle to 90 using the write() function. In this example, the robot is programmed to increment its position variables (xPos and yPos) by 100 after each machining task, but these values can be changed to perform more complex machining tasks

RESULTS:

This we have done the Robot Programming and simulation for machining (cutting, welding)

Post- Experiment Questions

1. What is the control algorithm for the robotic system?
2. What is the purpose of testing the robotic system?
3. What are some machining tasks that can be performed using the robotic system?

LAB EXPERIMENT 8

Robot Programming and simulation for writing practice

AIM:

To do the Robot Programming and simulation for writing practice

Equipment Required

1. Arduino board
2. USB cable
3. Breadboard
4. Jumper wires
5. Servo motor
6. Writing instrument (e.g. pen, pencil)
7. Autodesk Fusion 360 or MATLAB Simulink (optional for simulation)

Pre- Experiment Questions:

1. What is the purpose of this laboratory experiment
2. What is the importance of programming and simulation in robotics
3. What is the function of the Servo library in this experiment

The Arduino board, USB cable, breadboard, and jumper wires are required to connect the servo motor to the Arduino and program it using the Arduino IDE. The servo motor is used to control the writing instrument (e.g. pen or pencil) to perform writing tasks. Autodesk Fusion 360 or MATLAB Simulink can be used for simulation purposes to visualize and test the robot control algorithm before implementing it on a physical robot.

Arduino code:

```
#include <Servo.h> // Import the Servo library


// Define the servo pin
#define servoPin 9
// Create the Servo object
```

```
Servo servo;
// Define the position variables
int angle = 0;
void setup() {
  // Attach the servo to the pin
  servo.attach(servoPin);
}
void loop() {
  // Move the servo to the desired angle
  servo.write(angle);
  // Increment the angle variable (for a simple example)
  angle += 10;
  // Delay to allow the servo to move
  delay(500);
    }
```

This code uses the Servo library to control the servo motor for writing. The robot is moved to a desired angle by setting the angle using the write() function. In this example, the robot is programmed to increment its angle variable by 10 after each writing task, but these values can be changed to perform more complex writing tasks.

In addition to the programming code, you can also create a simulation environment using a software like Autodesk Fusion 360 or MATLAB Simulink to visualize and test your robot control algorithm before implementing it on a physical robot.

RESULTS:

Thus we have done the Robot Programming and simulation for writing practice

Post-Experiment Questions

1. How does the write() function control the servo motor
2. How can this code be modified to perform more complex writing tasks
3. What is the advantage of simulating the robot control algorithm before implementing it on a physical robot
4. What are the limitations of using Arduino for robotics

LAB EXPERIMENT 9

Robot Programming and simulation for any industrial process (Packaging, Assembly)

AIM:

To do the Robot Programming and simulation for any industrial process (Packaging, Assembly)

Equipment Required

1. Arduino board
2. USB cable
3. Breadboard
4. Jumper wires
5. Two Servo motors
6. Industrial equipment (e.g. packaging or assembly parts)
7. Autodesk Fusion 360 or MATLAB Simulink (optional for simulation)

Pre- Experiment Questions:

1. What is the purpose of this laboratory experiment?
2. How can robots be used in industrial processes such as packaging and assembly?
3. How does the Servo library in Arduino help control the servo motors used in this experiment?

The Arduino board, USB cable, breadboard, and jumper wires are required to connect the servo motors to the Arduino and program it using the Arduino IDE. The two servo motors are used to control the industrial equipment (e.g. packaging or assembly parts) to perform industrial processes. Autodesk Fusion 360 or MATLAB Simulink can be used for simulation purposes to visualize and test the robot control algorithm before implementing it on a physical robot.

Arduino Code:

```
#include <Servo.h> // Import the Servo library
```

```
// Define the servo pins

#define servoPin1 9

#define servoPin2 10

// Create the Servo objects

Servo servo1;

Servo servo2;

// Define the position variables

int angle1 = 0;

int angle2 = 0;

void setup() {

  // Attach the servos to the pins

  servo1.attach(servoPin1);

  servo2.attach(servoPin2);

}

void loop() {

  // Move the servos to the desired angles

  servo1.write(angle1);

  servo2.write(angle2);

  // Increment the angle variables (for a simple example)

  angle1 += 10;

  angle2 -= 10;

  // Delay to allow the servos to move

  delay(500);
      }
```

This code uses the Servo library to control two servo motors for industrial processes such as packaging or assembly. The robot is moved to desired angles by setting the angle using the write() function. In this example, the robot is programmed to increment its first angle variable by 10 and decrement its second angle variable by 10 after each cycle, but these values can be changed to perform more complex industrial processes.

In addition to the programming code, you can also create a simulation environment using a software like Autodesk Fusion 360 or MATLAB Simulink to visualize and test your robot control algorithm before implementing it on a physical robot.

RESULTS:

Thus we have done the Robot Programming and simulation for any industrial process
(Packaging, Assembly)

Post-Experiment Questions

1. How does the write() function control the servo motors in this experiment?
2. How can the code be modified to perform different industrial processes?
3. What are the advantages of simulating the robot control algorithm before implementing it on a physical robot?
4. What are the limitations of using Arduino for industrial robotics?
5. How can sensors be used in conjunction with this code to improve the accuracy and efficiency of the industrial process being performed?
6. What safety measures should be taken when working with industrial robots in a laboratory or industrial setting?
7. What is the future of industrial robotics and automation, and how might it impact the workforce?

LAB EXPERIMENT 10

Robot Programming and simulation for multi process

AIM:

To do the Robot Programming and simulation for multi process

Equipment Required

- Arduino board
- USB cable
- Breadboard
- Jumper wires
- Two Servo motors
- Industrial equipment (e.g. packaging or assembly parts)
- Autodesk Fusion 360 or MATLAB Simulink (optional for simulation)

Pre-Experiment Questions

1. What is the purpose of this laboratory experiment?
2. How can robots be used for multiple processes in industrial automation?
3. How does the Servo library in Arduino help control the servo motors used in this experiment?
4. What safety measures should be taken when working with industrial robots in a laboratory or industrial setting?

The Arduino board, USB cable, breadboard, and jumper wires are required to connect the servo motors to the Arduino and program it using the Arduino IDE. The two servo motors are used to control the tools / equipment for multiple processes. Autodesk Fusion 360 or MATLAB Simulink can be used for simulation purposes to visualize and test the robot control algorithm before implementing it on a physical robot.

Arduino code:

```
#include <Servo.h> // Import the Servo library
// Define the servo pins
#define servoPin1 9
#define servoPin2 10
// Create the Servo objects
Servo servo1;
Servo servo2;
// Define the position variables
```

```
int angle1 = 0;
int angle2 = 0;
void setup() {
  // Attach the servos to the pins
  servo1.attach(servoPin1);
  servo2.attach(servoPin2);
}
void loop() {
  // Move the servos to the desired angles
  servo1.write(angle1);
  servo2.write(angle2);

  // Increment the angle variables (for a simple example)
  angle1 += 10;
  angle2 -= 10;
  // Delay to allow the servos to move
  delay(500);

  // Switch to a new process after a certain number of cycles
  if (angle1 == 180 && angle2 == -180) {
    angle1 = 0;
    angle2 = 0;
    delay(1000);
  }
}
```

This code uses the Servo library to control two servo motors for multiple processes. The robot is moved to desired angles by setting the angle using the write() function. In this example, the robot is programmed to increment its first angle variable by 10 and decrement its second angle variable by 10 after each cycle until it reaches 180 and -180 respectively, at which point it switches to a new process by resetting the angle variables to 0 and waiting for 1 second.

In addition to the programming code, you can also create a simulation environment using a software like Autodesk Fusion 360 or MATLAB Simulink to visualize and test your robot control algorithm before implementing it on a physical robot.

RESULTS:

Thus we have done the Robot Programming and simulation for multi process

Post-Experiment Questions

1. How does the write() function control the servo motors in this experiment?
2. How can the code be modified to perform different multiple processes?
3. What are the advantages of simulating the robot control algorithm before implementing it on a physical robot?
4. What are the limitations of using Arduino for controlling multiple processes in industrial automation?
5. How can sensors be used in conjunction with this code to improve the accuracy and efficiency of the industrial process being performed?

This lab manual has been updated by

Dr. R. Dheivanai

(r.dheivanai@ggnindia.dronacharya.info)

Cross checked by

HoD / EEE & ECE