SWAMI KESHVANAND INSTITUTE OF TECHNOLOGY,
MANAGEMENT AND GRAMOTHAN, JAIPUR

Hands on Lab Guide
(Lab Manual)

ROBOT PROGRAMMING LAB
IV Year B.Tech VIII SEM
(Course Code: 8CAI4-22)
Session 2023-2024

Department of Computer Science & Engineering (AI)
SKIT, JAIPUR

Swami Keshvanand Institute of Technology, Management & Gramothan, Ramnagaria, Jagatpura, Jaipur-302017

LAB MANUAL
Robot Programming Lab (8CAI4-22)

VERSION 1.0

	AUTHOR / OWNER	REVIEWED BY	APPROVED BY
NAME	Dr. Sarabjeet Singh		Dr. Mehul Mahrishi
DESIGNATION	Associate Professor		HOD (CSE)
SIGNATURE & REVIEW DATE			
SIGNATURE & REVIEW DATE			
SIGNATURE & REVIEW DATE			

Department of Computer Science & Engineering (AI)
Jaipur – 302017, Rajasthan (INDIA)
E-Mail: hodcs@skit.ac.in, URL: www.skit.ac.in

GENERAL LABORATORY RULES

Responsibilities of Students

Students are expected to follow some fairly obvious rules of conduct:

DO's:

- Enter the lab on time and leave at the proper time.
- Wait for the previous class to leave before the next class enters.
- Keep the bag outside on the respective racks.
- Turn off the machine before leaving the lab.
- Leave the labs at least as nice as you found them.
- If you notice a problem with a piece of equipment (e.g., a computer doesn't respond) or the room in general (e.g., cooling, heating, lighting) please report it to lab assistant immediately.
- Please be considerate of those around you, especially in terms of noise level. While labs are a natural place for conversations of all types, kindly keep the volume turned down.

DON'Ts:

- Do not misuse the equipment.
- Do not adjust the heat or air conditioners. If you feel the temperature is not properly set, inform lab staff; we will attempt to maintain a balance that is healthy for people and machines.
- Do not attempt to reboot the computer. Report the problems to lab assistant.
- Do not remove or modify any software or file without permission.
- Do not remove printers and machines from the network without the permission of lab assistant.
- Do not monopolize equipment. If you're going to be away from your machine for more than 10 or 15 minutes, log out before leaving. This is both for the security of your account, and to ensure that others are able to use the lab resources while you are not.
- Playing games is not allowed in the lab.
- No hardware including USB drives can be connected or disconnected in the labs without prior permission of the lab assistant.
- Eatables are not allowed in the lab.
- Don't bring the mobile phones in the lab. If necessary then keep them in silence mode.

INSTRUCTIONS

BEFORE ENTERING IN THE LAB

- All the students are supposed to prepare the theory regarding the next experiment/ Program.
- Students are supposed to bring their lab records as per their lab schedule.
- Previous experiment/program should be written in the lab record.
- If applicable trace paper/graph paper must be pasted in lab record with proper labeling.
- Students must follow the instructions, failing which he/she may not be allowed in the lab.

WHILE WORKING IN THE LAB

- Adhere to experimental schedule as instructed by the faculty.
- Get the previously performed experiment/ program signed by the faculty.
- Get the output of current experiment/program checked by the faculty in the lab copy.
- Each student should work on his/her assigned computer at each turn of the lab.
- Take responsibility of valuable accessories.
- If anyone is caught red-handed carrying any equipment of the lab, then he/she will have to face serious consequences.

Marking/Assessment System

Total Marks -100

Distribution of Marks - 60 (Sessional)

Attendance	File Work	Performance	Viva	Total
10	10	30	10	60

Distribution of Marks - 40 (End Term) Depends on the Examiner

File Work	Performance	Viva	Total
10	20	10	40

RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Syllabus

IV Year-VIII Semester: B.Tech. Computer Science and Engineering (AI)

8CAI4-22: Robot Programming Lab

Credit: 1 **Max. Marks: 100(IA:60, ETE:40)**

0L+0T+2P **End Term Exam: 2 Hours**

SN	List of Experiments
1	An introduction to robot programming.
2	**Object Detection and Tracking Robot:** Create a robot with a camera that can detect and track objects in its field of view. Implement object detection algorithms and use them for tracking and interaction.
3	**Autonomous Maze Solving Robot:** Construct a robot that can autonomously navigate through a maze from the start to the finish. Implement maze-solving algorithms like A* or Dijkstra's algorithm.
4	**Reinforcement Learning for Robotic Arm Control:** Train a robotic arm to perform tasks using reinforcement learning. Implement algorithms like Deep Q-Networks (DQN) or Proximal Policy Optimization (PPO) to optimize arm movements.
5	**Human-Robot Interaction using Natural Language Processing (NLP):** Design a robot that can understand and respond to voice commands. Use NLP techniques to process and interpret human language to control the robot's actions.
6	**Robot-Assisted Healthcare and Patient Interaction:** Design a robot that can assist patients and healthcare professionals. Use AI to understand patient needs, provide information, and interact in a helpful and empathetic manner.
7	**Gesture Recognition and Control of Robotic Arm:** Build a robotic arm that responds to hand gestures. Train a machine learning model to recognize gestures, and use them to control the movements of the robotic arm.
8	**Obstacle Avoidance Robot with Ultrasonic Sensors:** Develop a robot capable of navigating an environment while avoiding obstacles using ultrasonic sensors. Implement basic obstacle avoidance algorithms and refine the robot's movements.

Beyond Syllabus:

1. Interface Temperature & Humidity sensor using Raspberry pi and send the data over ThingSpeak cloud Infrastructure.
2. Interface Ultrasonic sensor with Raspberry pi to measure distance.
3. Design a system using Arduino to monitor Temperature and Humidity of a place on an IDE/LCD.

INDEX

LAB PLAN

Total number of experiment 11

Total number of turns required 11

Number of turns required for

Experiment Number	Turns	Scheduled Day
Exp. 1	1	Day 1
Exp. 2	1	Day 2
Exp. 3	1	Day 3
Exp. 4	1	Day 4
Exp. 5	1	Day 5
Exp. 6	1	Day 6
Exp. 7	1	Day 7
Exp. 8	1	Day 8
Exp. 9	1	Day 9
Exp. 10	1	Day 10
Exp. 11	1	Day 11

Distribution of Lab Hours:

Attendance	05 minutes
Explanation of features of language	15 minutes
Explanation of experiment	15 minutes
Performance of experiment	70 minutes
Viva / Quiz / Queries	15 minutes
Total	**120 Minutes (2 Hrs.)**

Lab Objectives and Outcome

Objectives

The objective of this course is to impart necessary and practical knowledge of components employed in Robotics and develop skills required to build real-life Robotics projects.

- Teach students the principles of how robots move and interact with their environment, covering forward and inverse kinematics, velocity, acceleration, and torque.
- Develop students' proficiency in robotics programming languages like Python, C++, and ROS, enabling them to control robot behaviors effectively.
- Familiarize students with robotics sensors like cameras, LiDAR, ultrasonic sensors, teaching integration to understand the robot's surroundings.
- Teach students techniques for planning efficient robot paths in complex environments, using algorithms like A*, RRT, and PID controllers for motion control.
- Offer students hands-on projects and challenges to apply their robotics knowledge to real-world applications like autonomous navigation, robotic manipulation, human-robot interaction, and swarm robotics.

Course Outcomes

After completion of this course, students will be able to –

8CAI4-22.1	**Associate** a thorough understanding of robot programming concepts, including but not limited to motion control, sensor integration, and decision-making algorithms.
8CAI4-22.1	**Demonstrate** and troubleshoot complex behaviors for robots using programming languages such as Python or C++, integrating sensory input to make decisions and perform tasks autonomously.
8CAI4-22.1	**Apply** software engineering principles such as modular design, version control, and documentation to develop robust and maintainable software for controlling robots.
8CAI4-22.1	**Analyze** real-world problems that can be addressed through robotics, propose innovative solutions, and implement these solutions by programming robots to perform specific tasks or functions.
8CAI4-22.1	**Program** and test robot systems to achieve specified objectives.

Experiment – 1

Aim: An introduction to robot programming.

Introduction:

This course delves into the exciting world of robotics, learning how to program robots to perform various tasks efficiently and effectively. This laboratory manual serves as a guide for understanding the fundamentals of robot programming, exploring key concepts, and gaining hands-on experience with programming robots.

Objective:

The objective of this laboratory session is to provide a comprehensive understanding of robot programming principles & techniques. By the end of this session, students should be able to:

- Understand the basics of robot programming languages.
- Familiarize with the common components of a robot programming environment.
- Develop essential skills in programming robots to perform specific tasks.
- Gain practical experience through hands-on exercises and projects.

Prerequisites:

Before proceeding with this laboratory session, it is assumed that students have a basic understanding of:

- Programming fundamentals, including variables, control structures, and functions.
- Mathematics, particularly algebra and geometry.
- Familiarity with basic robotics concepts such as sensors, actuators, and kinematics.

Equipment and Software:

For this laboratory session, following equipment and software are needed:

- **Robot platform:** This could be a physical robot or a simulator depending on the availability and requirements of the laboratory setup.

- **Programming environment:** A software environment where you can write, compile, and execute robot programs. Common examples include ROS (Robot Operating System), MATLAB Robotics Toolbox, or proprietary software provided by the robot manufacturer.
- **Computer with internet access:** To download necessary software, access online resources, and communicate with instructors or peers if required.

Laboratory Activities:

The laboratory session will consist of the following activities:

- **Introduction to Robot Programming Languages:** Students will explore different programming languages commonly used in robotics, such as Python, C++, and ROS-specific languages. Understanding the syntax and features of these languages is essential for effective robot programming.
- **Setting up the Programming Environment:** Students will learn how to set up the programming environment, including installing necessary software, configuring communication with the robot platform, and setting up simulation environments if applicable.
- **Basic Robot Control:** Students will start with simple exercises to control the robot's motion, including moving forward, backward, turning, and stopping. These exercises will help students understand how to send commands to the robot and control its behavior.
- **Sensor Integration:** Sensors play a crucial role in robotics by providing feedback about the robot's environment. Students will learn how to integrate sensors such as proximity sensors, cameras, and encoders into your robot programs to enable more sophisticated behaviors.
- **Path Planning and Navigation:** Students will delve into algorithms for path planning and navigation, allowing the robot to autonomously navigate its environment while avoiding obstacles. Concepts such as localization, mapping, and obstacle avoidance will be covered.
- **Advanced Topics (Optional):** Depending on the time and resources available, students may explore advanced topics such as robot manipulation, vision-based control, machine learning for robotics, or collaborative robotics.

Conclusion:

Robot programming is a fascinating field that combines elements of computer science, engineering, and mathematics. Through this laboratory session, students will gain valuable hands-on experience and develop the skills necessary to program robots for a wide range of applications.

Experiment – 2

Aim: Object Detection and Tracking Robot:

In this example, we'll use the MobileNet-SSD (Single Shot MultiBox Detector) model, which is a popular object detection model that's capable of detecting multiple objects in an image. First, you'll need to install the necessary libraries. You can do this with pip:

```
pip install opencv-python
pip install imutils
```

Code:

Here's a simple example of object detection using OpenCV and MobileNet-SSD:

```
import cv2
import imutils

# Load pre-trained model
net = cv2.dnn.readNetFromCaffe('MobileNetSSD_deploy.prototxt.txt',
'MobileNetSSD_deploy.caffemodel')

# Initialize video stream
vs = cv2.VideoCapture(0)
```

```
while True:
    # Read the next frame from the video stream
    _, frame = vs.read()

    # Resize the frame to have a maximum width of 400 pixels
    frame = imutils.resize(frame, width=400)

    # Convert the frame to a blob
    blob = cv2.dnn.blobFromImage(frame, 0.007843, (300, 300), 127.5)

    # Pass the blob through the network and obtain the detections
    net.setInput(blob)
    detections = net.forward()

    # Loop over the detections
    for i in np.arange(0, detections.shape[2]):
        # Extract the confidence (i.e., probability) associated with the prediction
        confidence = detections[0, 0, i, 2]

        # Filter out weak detections by ensuring the `confidence` is greater than the minimum confidence
        if confidence > 0.2:
            # Extract the index of the class label from the `detections`
            idx = int(detections[0, 0, i, 1])

            # Draw the prediction on the frame
            label = "{}: {:.2f}%".format(CLASSES[idx], confidence * 100)
            cv2.putText(frame, label, (startX, startY-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

    # Show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # If the `q` key was pressed, break from the loop
    if key == ord("q"):
        break
```

```
# Clean up
cv2.destroyAllWindows()
vs.stop()
```

Experiment – 3

Aim: Autonomous Maze Solving Robot:

The A algorithm is a popular path finding algorithm used in many applications, including games and robotics. It works by maintaining a tree of paths originating at the start node and extending those paths one edge at a time until its termination criterion is satisfied.

Code:

```
import heapq

def heuristic(a, b):
    return abs(b[0] - a[0]) + abs(b[1] - a[1])

def astar(array, start, goal):
    neighbors = [(0,1),(0,-1),(1,0),(-1,0)]
    close_set = set()
    came_from = {}
    gscore = {start:0}
    fscore = {start:heuristic(start, goal)}
    oheap = []

    heapq.heappush(oheap, (fscore[start], start))
```

```
  while oheap:
    current = heapq.heappop(oheap)[1]

    if current == goal:
      data = []
      while current in came_from:
        data.append(current)
        current = came_from[current]
      return data

    close_set.add(current)
    for i, j in neighbors:
      neighbor = current[0] + i, current[1] + j
      tentative_g_score = gscore[current] + heuristic(current, neighbor)
      if 0 <= neighbor[0] < array.shape[0]:
        if 0 <= neighbor[1] < array.shape[1]:
          if array[neighbor[0]][neighbor[1]] == 1:
            continue
        else:
          # array bound y walls
          continue
      else:
        # array bound x walls
        continue

      if neighbor in close_set and tentative_g_score >= gscore.get(neighbor, 0):
        continue

      if  tentative_g_score < gscore.get(neighbor, 0) or neighbor not in [i[1]for i in
oheap]:
        came_from[neighbor] = current
        gscore[neighbor] = tentative_g_score
        fscore[neighbor] = tentative_g_score + heuristic(neighbor, goal)
        heapq.heappush(oheap, (fscore[neighbor], neighbor))

  return False

# Define the maze - 0 is open, 1 is blocked
maze = [[0, 0, 0, 0, 1, 0],
```

```
    [1, 1, 0, 0, 1, 0],
    [0, 0, 0, 1, 0, 0],
    [0, 1, 1, 0, 0, 1],
    [0, 0, 0, 0, 1, 0]]

start = (0, 0)  # Starting position
end = (4, 5)  # Ending position

path = astar(maze, start, end)
print(path)
```

Experiment – 4

Aim: Reinforcement Learning for Robotic Arm Control:

Training a robotic arm to perform tasks using reinforcement learning involves complex simulations and hardware setups. Here's a simplified Python code snippet using the OpenAI Gym environment and the Proximal Policy Optimization (PPO) algorithm to train a robotic arm in a simulated
environment:

```
import gym
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow_probability import distributions

# Define the environment
env = gym.make('Pendulum-v0')

# Define the neural network model
```

```
model = Sequential([
  Dense(64, activation='relu', input_shape=(env.observation_space.shape[0],)),
  Dense(64, activation='relu'),
  Dense(env.action_space.shape[0])
])

# Define the optimizer
optimizer = Adam(learning_rate=0.001)

# Define the PPO agent
def ppo_agent(model, optimizer):
  action_space = env.action_space.shape[0]

  def agent(observation):
    observation = np.expand_dims(observation, axis=0)
    action_logits = model(observation)
    action_distribution = distributions.Categorical(logits=action_logits)
    action = action_distribution.sample()
    return action.numpy()[0]

  return agent

# Train the agent using PPO
agent = ppo_agent(model, optimizer)
for episode in range(1000):
  observation = env.reset()
  done = False
  while not done:
    action = agent(observation)
    next_observation, reward, done, _ = env.step(action)
    observation = next_observation

# Test the trained agent
observation = env.reset()
done = False
while not done:
  action = agent(observation)
  observation, reward, done, _ = env.step(action)
  env.render()
```

```
env.close()
```

Experiment 5

Aim: Human-Robot Interaction using Natural Language Processing (NLP):

Designing a robot that can understand and respond to voice commands involves integrating
speech recognition and natural language processing (NLP) techniques. Here's a simplified
Python code snippet using the SpeechRecognition library and the Natural Language Toolkit to process voice commands and control the robot's actions:

Code:

```
import speech_recognition as sr
import nltk
from nltk.tokenize import word_tokenize

# Initialize the speech recognizer
recognizer = sr.Recognizer()

# Function to process voice commands
def process_voice_command():
```

```
    with sr.Microphone() as source:
        print("Listening...")
        audio = recognizer.listen(source)

    try:
        command = recognizer.recognize_google(audio)
        print("You said:", command)
        return command
    except sr.UnknownValueError:
        print("Sorry, I could not understand the command.")
        return ""
    except sr.RequestError:
        print("Sorry, my speech service is down.")
        return ""

# Function to interpret and respond to voice commands
def interpret_and_respond(command):
    tokens = word_tokenize(command)

    if 'move' in tokens:
        direction = tokens[tokens.index('move') + 1]
        if direction == 'forward':
            print("Moving the robot forward.")
        elif direction == 'backward':
            print("Moving the robot backward.")
        else:
            print("Invalid direction command.")
    else:
        print("Command not recognized.")

# Main loop to listen for voice commands
while True:
    command = process_voice_command()
    if command:
        interpret_and_respond(command)
```

Experiment 6

Aim: Robot-Assisted Healthcare and Patient Interaction:

Designing a robot to assist patients and healthcare professionals involves integrating AI tec hnologiesfor understanding patient needs, providing information, and interacting empathet ically. This Python code snippet using the ChatterBot library to create a chatbot that interact with patients and

healthcare professionals:

In this code:

- We use the ChatterBot library to create a chatbot instance named 'HealthcareBot'.
- We train the chatbot on the English language corpus to provide responses to a wide ran ge of queries.
- The chat_with_bot() function allows users to interact with the chatbot by entering text i nputs.
- The chatbot responds to user inputs based on the training data and the conversational p atterns.

Code:

```
from chatterbot import ChatBot
from chatterbot.trainers import ChatterBotCorpusTrainer

# Create a ChatBot instance
chatbot = ChatBot('HealthcareBot')

# Create a new trainer for the chatbot
trainer = ChatterBotCorpusTrainer(chatbot)

# Train the chatbot on the English language corpus
trainer.train("chatterbot.corpus.english")

# Function to interact with the chatbot
def chat_with_bot():
    print("HealthcareBot: Hello, how can I assist you today?")
    while True:
        user_input = input("You: ")
        if user_input.lower() == 'exit':
            print("HealthcareBot: Goodbye!")
            break
        response = chatbot.get_response(user_input)
        print("HealthcareBot:", response)

# Start interacting with the chatbot
chat_with_bot()
```

Experiment – 7

Aim: Gesture Recognition and Control of Robotic Arm:

Building a robotic arm that responds to hand gestures involves training a machine learni ng model to recognize gestures and mapping them to specific robotic arm movements. Here's a simplified Python code snippet using the MediaPipe library for hand gesture rec ognition and controlling a robotic arm:

```
import mediapipe as mp
import cv2

# Initialize MediaPipe Hands
mp_hands = mp.solutions.hands
hands = mp_hands.Hands()

# Initialize the robotic arm (simulated)
class RoboticArm:
    def move_up(self):
        print("Moving the robotic arm up")
```

```
    def move_down(self):
        print("Moving the robotic arm down")

# Create an instance of the robotic arm
robotic_arm = RoboticArm()

# Function to control the robotic arm based on hand gestures
def control_robotic_arm(hand_landmarks):

    # Extract hand landmarks for gesture recognition
    # Implement your gesture recognition logic here

    # Example: If the hand is open, move the arm up; if the hand is closed, move the arm
down
    if hand_landmarks:
        if hand_landmarks[8].y > hand_landmarks[5].y:
            robotic_arm.move_up()
        else:
            robotic_arm.move_down()

# Open a video stream
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # Process the frame to detect hand landmarks
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = hands.process(frame_rgb)

    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            control_robotic_arm(hand_landmarks.landmark)

    cv2.imshow('Robotic Arm Control', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the video stream and close all windows
cap.release()
cv2.destroyAllWindows()
```

In this code:

- MediaPipe library is used for hand gesture recognition to detect hand landmarks.
- A robotic arm is simulated with simple movements (up and down).
- The control_robotic_arm() function interprets hand gestures to control the robotic arm movements.
- The robotic arm moves up if the hand is open and down if the hand is closed.
- The code captures video frames from the camera, processes them for hand gesture recognition, and controls the robotic arm based on the detected gestures.

Experiment 8

Aim: Obstacle Avoidance Robot with Ultrasonic Sensors:

Developing a robot capable of navigating an environment while avoiding obstacles using ultrasonic sensors involves integrating sensor data with obstacle avoidance algorithms. Here's a simplified Python code snippet using simulated sensor data and basic obstacle avoidance logic:

```
import random

class Robot:
    def __init__(self):
        self.position = [0, 0]  # Initial position of the robot
        self.obstacle_threshold = 5  # Distance threshold to detect obstacles
        self.max_speed = 1  # Maximum speed of the robot

    def move(self, direction, distance):
```

```
        if direction == 'forward':
            self.position[0] += distance
        elif direction == 'backward':
            self.position[0] -= distance
        elif direction == 'right':
            self.position[1] += distance
        elif direction == 'left':
            self.position[1] -= distance

    def avoid_obstacle(self):
        obstacle_detected = random.choice([True, False])  # Simulated obstacle detection
        if obstacle_detected:
            print("Obstacle detected! Avoiding obstacle...")
            self.move('backward', 1)  # Move backward to avoid the obstacle
        else:
            print("No obstacle detected. Continuing forward...")
            self.move('forward', self.max_speed)  # Move forward

# Create an instance of the robot
robot = Robot()

# Simulate robot navigation with obstacle avoidance
for _ in range(10):  # Simulate 10 steps of movement
    robot.avoid_obstacle()
    print("Current Position:", robot.position)
```

In this code:

- We define a Robot class with methods to move the robot and avoid obstacles.
- The move() method updates the robot's position based on the specified direction and distance.
- The avoid_obstacle() method simulates obstacle detection using random values and Moves the robot backward if an obstacle is detected.
- We create an instance of the robot and simulate its navigation for 10 steps, printing the current position at each step.

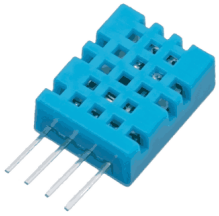
Beyond Syllabus

Experiment – 9

Aim: Interface Temperature & Humidity sensor using Raspberry pi and send the data over ThingSpeak cloud Infrastructure.

Components: Raspberry pi, DHT11/DHT22 sensor.

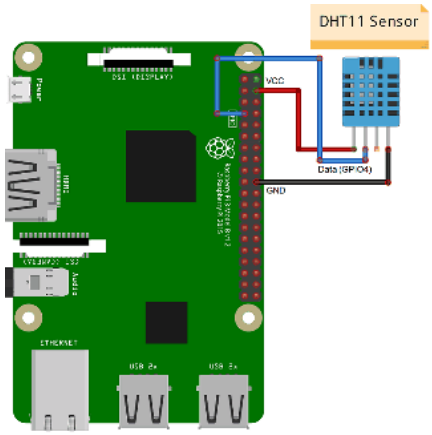
Explanation:



DHT11 Sensor

- DHT11 sensor measures humidity and temperature values serially over a single wire.
- It can measure relative humidity in percentage (20 to 90% RH) and temperature in degree Celsius in the range of 0 to 50°C.
- It has 4 pins; one of which is used for data communication in serial form.
- Pulses of different TON and TOFF are decoded as logic 1 or logic 0 or start pulse or end of a frame.

Circuit:



Code:

- Here, we are going to interface DHT11 sensor with Raspberry Pi 3 and display Humidity and Temperature on terminal.
- We will be using the DHT Sensor Python library by Adafruit from GitHub. The Adafruit Python DHT Sensor library is created to read the Humidity and Temperature on raspberry Pi or Beaglebone Black. It is developed for DHT series sensors like DHT11, DHT22 or AM2302.

- https://github.com/adafruit/Adafruit_Python_DHT/archive/master.zip

  Extract the library and install it in the same root directory of downloaded library by executing following command,

```
sudo python setup.py install
```

- Once the library and its dependencies has been installed, open the example sketch named simple test from the library kept in examples folder.
- In this code, raspberry Pi reads Humidity and Temperature from DHT11 sensor and prints them on terminal. But, it read and display the value only once. So, here we made change in the program to print value continuously.

Note:

- Assign proper sensor type to the sensor variable in this library. Here, we are using DHT11 sensor.

```
sensor = Adafruit_DHT.DHT11
```

- If anyone is using sensor DHT22 then we need to assign **Adafruit_DHT.DHT22** to the sensor variable shown above.
- Then assign pin no. to which DHT sensor's data pin is connected. Here, data out of DHT11 sensor is connected to GPIO4. As shown in above interfacing diagram.

```
sudo apt-get install http.client
```

```
#Code:
import Adafruit_DHT
import http.client
import urllib
import time
key = "CKUNX8CN1DHO6MO0"  # Put your API Key here

# Sensor should be set to Adafruit_DHT.DHT11,
# Adafruit_DHT.DHT22, or Adafruit_DHT.AM2302.
sensor = Adafruit_DHT.AM2302

# Example using a Raspberry Pi with DHT sensor
# connected to GPIO4.
pin = 4
```

```
def thermometer():
  while True:
    #Calculate CPU temperature of Raspberry Pi in Degrees C
    humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
  #print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
    if humidity is not None and temperature is not None:
      print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
    else:
      print('Failed to get reading. Try again!')
    params = urllib.parse.urlencode({'field1': temperature, 'field2': humidity, 'key':key })
    headers = {"Content-typZZe": "application/x-www-form-urlencoded","Accept": "text/plain"}
    conn = http.client.HTTPConnection("api.thingspeak.com:80")
    try:
conn.request("POST", "/update", params, headers)
      response = conn.getresponse()
      #print (temp)
      print (response.status, response.reason)
      data = response.read()
conn.close()
    except:
      print ("connection failed")
    break
if __name__ == "__main__":
  while True:
    thermometer()
```

ThingSpeak Cloud:

ThingSpeak is an open IoT platform for monitoring the data online. In ThingSpeak, the data in the channel can be set as private or public according to the choice. ThingSpeak takes minimum of 15 seconds to update the readings. It's a great and very easy to use platform for building IOT projects.

The first step in building any IoT project using Raspberry Pi is to upload the data to any cloud server using Raspberry Pi. In this simplest Raspberry Pi IOT project, ThingSpeak is used as a cloud server to store the data. Here Raspberry Pi will read DHT11/DHT22 sensor data and send it to ThingSpeak. The sensed data can be monitored from anywhere in the world via internet. This will be useful for some application where Pi is running for long time at some remote place and CPU temperature needs to be monitored.

Step 1: Signup for ThingSpeak

For creating the channel on ThingSpeak you first need to sign up on ThingSpeak. In case if you already have account on ThingSpeak just sign in using your id and password.

For creating your account go to www.thingspeak.com

Step 2: Create a Channel for Your Data

Once Signing is done, create a new channel by clicking “New Channel” button

Step 3: Getting API Key in ThingSpeak

To send data to ThingSpeak, a unique API key is needed, which will be used later in the python code to upload the CPU data onto ThingSpeak Website.

Click on “API Keys” button to get the unique API key for uploading the CPU data.

Step 4: Python Code for Raspberry Pi

Run the script that is mentioned on previous page

Step 5: Check ThingSpeak site for Data Logging

After completing these steps, open the channel and observe that the DHT sensor data is getting updated into ThingSpeak website.

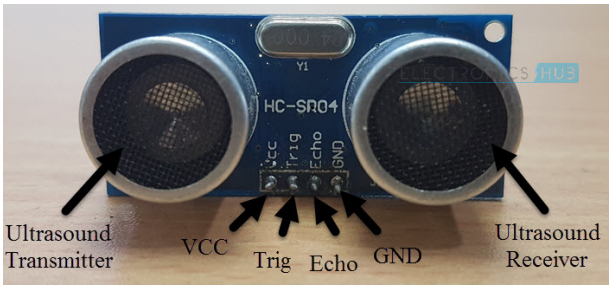
Experiment – 10

Aim: Interface Ultrasonic sensor with Raspberry pi to measure distance.

Components: Ultrasonic Sensor (HC-SR04), Raspberry Pi, Resistor

Explanation:

Ultrasonic Sensors, particularly HC-SR04 Ultrasonic Sensor, are very popular among electronic hobbyists and are frequently used in a variety of projects like Obstacle Avoiding Robot, Distance Measurement, Proximity Detection and so forth. In this project, we will learn about HC-SR04 Ultrasonic and see how to interface one with Raspberry Pi.



We will now see how to measure the distance of an object using HC-SR04 Ultrasonic Sensor. In order to send the 40 KHz Ultrasound, the TRIG Pin of the Ultrasonic Sensor must be held HIGH for a minimum duration of 10µS.After this, the Ultrasonic Transmitter, will transmits a burst of 8-pulses of ultrasound at 40 KHz. Immediately, the control circuit in the sensor will change the state of the ECHO pin to HIGH. This pin stays HIGH until the ultrasound hits an object and returns to the Ultrasonic Receiver. Based on the Time for which the Echo Pin stays HIGH, you can calculate the distance between the sensor and the object. For example, if we calculated the time for which ECHO is HIGH as 588µS, then you can calculate the distance with the help of the speed of sound, which is equal to 340m/s.

Distance = Velocity of Sound / (Time/2) = 340m/s / (588µS /2) = 10cm.

Explanation:

Physical Connections:-

In breadboard connect ultrasonic sensor

Back side pins of sensor:- (in breadboard vertical connections(in between) are there)

1) VCC to 5v in Arduino board
2) Trig to pin 13 of digital pin of Arduino
3) Echo to pin 12 of digital pin of Arduino
4) Gnd to Gnd of Arduino board

Connect USB from computer to Arduino

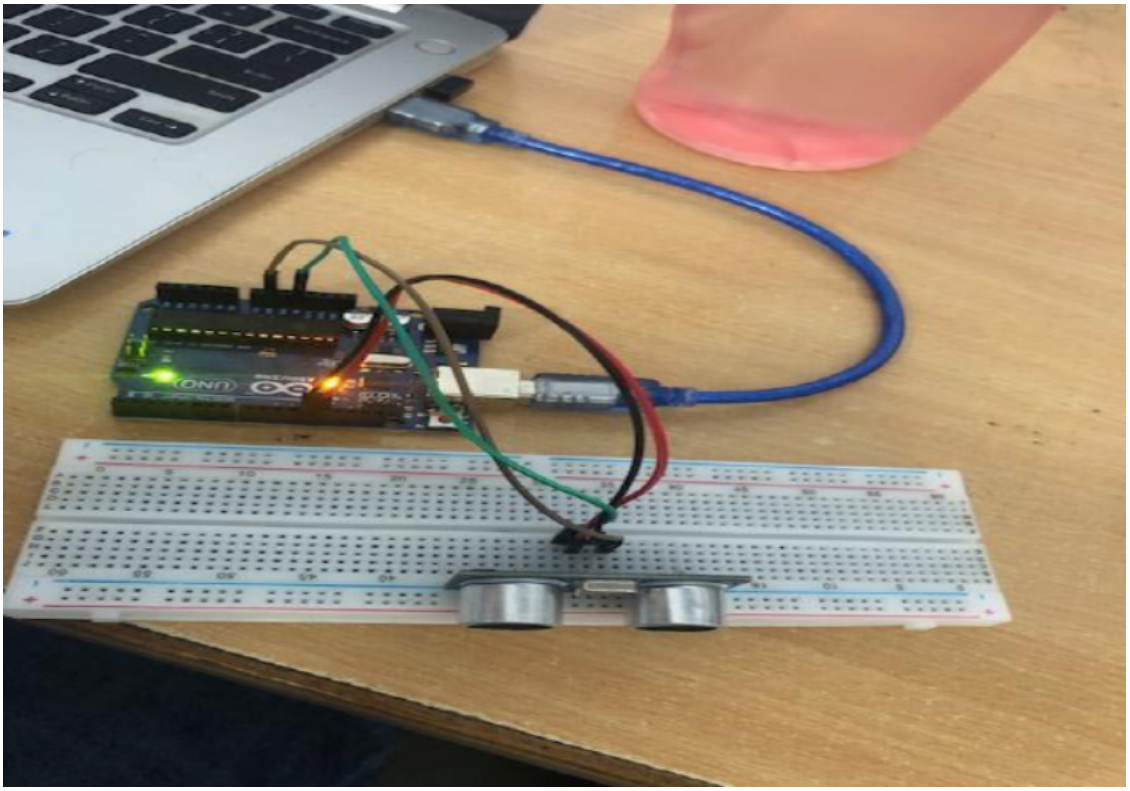
Code:

```
const int trigPin
= 13; const int
echoPin = 12;
void setup() {
Serial.begin(9
600);}  void
loop()
{
long duration, inches, cm;
pinMode(trigPin, OUTPUT);
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
pinMode(echoPin, INPUT);
duration = pulseIn(echoPin, HIGH);
inches = microsecondsToInches(duration);
cm = microsecondsToCentimeters(duration);
Serial.print(inches);
Serial.print("in, ");
Serial.print(cm);
```

```
Serial.print("cm");

Serial.println();

delay(1000);

}

long microsecondsToInches(long microseconds)

{return microseconds / 74 / 2;

}

long microsecondsToCentimeters(long microseconds)

{return microseconds / 29 / 2;}
```

Verify your code in Arduino IDE that is sketch. Upload your code in Arduino IDE. After connecting, all done. Press Ctrl + shift + M. You can see output window

Output:



COM3

```
77in, 198cm
77in, 196cm
907in, 2316cm
77in, 197cm
76in, 196cm
77in, 198cm
77in, 198cm
77in, 198cm
78in, 201cm
8in, 22cm
7in, 19cm
8in, 21cm
5in, 13cm
```

Autoscroll Show timestamp

Experiment – 11

Aim: Design a system using Arduino to monitor Temperature and Humidity of a place on an IDE/LCD.

Hardware Library: ARDUINO UNO

Simulator: Proteus

Components: Add Peripheral ->Groov -> Temp. And Humidity Sensor

Theory:

DHT11 Sensor:

The DHT11 is a basic, low cost digital temperature and humidity sensor.

- DHT11 is a single wire digital humidity and temperature sensor, which provides humidity and temperature values serially with one-wire protocol.
- DHT11 sensor provides relative humidity value in percentage (20 to 90% RH) and temperature values in degree Celsius (0 to 50 °C).
- DHT11 sensor uses resistive humidity measurement component, and NTC temperature measurement component.

Explanation:

Physical Connections:-

In breadboard connect DHT11 sensor

Back side pins of sensor: - (in breadboard vertical connections (in between) are there)

1) VCC to 5v in Arduino board
2) DATA to pin 4 of digital pin of Arduino
3) Gnd to Gnd of Arduino board

Connect USB from laptop to Arduino

Code:

```
#include <dht11.h>
#define DHT11PIN 4
dht11 DHT11;
```

```
void setup()
{
Serial.begin(9600);
}
void loop()
{
Serial.println();
 int chk = DHT11.read(DHT11PIN);
Serial.print("Humidity (%): ");
Serial.println((float)DHT11.humidity, 2);
Serial.print("Temperature (C): ");
Serial.println((float)DHT11.temperature,
2);
delay(2
000); }
```

Viva Questions

1) What is Robot Programming?

Robot programming involves creating instructions and algorithms that enable robots to perform specific tasks autonomously or semi-autonomously. These instructions can include movements, interactions with the environment, decision-making processes, and responses to various inputs from sensors. Robot programming can encompass a wide range of techniques and technologies, from low-level control of actuators to high-level task planning and decision-making algorithms. It often involves proficiency in programming languages such as Python, C++, or specialized robot programming languages like ROS (Robot Operating System).

2) Explain Raspberry Pi

Raspberry Pi is a computer which is capable of doing all the operations like a conventional computer. It has other features such as onboard WiFi, GPIO pins, and Bluetooth in order to communicate with external things.

3) How to run Raspberry pi in headless mode?

Raspberry pi in headless mode can be run by using SSH. The latest operating system has an inbuilt VNC server that is installed for taking remote desktop on Raspberry Pi.

4) What are the fundamental components of Robot Programming?

The fundamental components of robot programming include:

- **Motion Control:** Algorithms and commands that control the movement of robotic actuators such as motors or servos, enabling the robot to navigate, manipulate objects, or interact with its environment.
- **Sensor Integration:** Techniques for interfacing and integrating data from various sensors (e.g., cameras, LiDAR, proximity sensors) to perceive and understand the robot's surroundings, including object detection, localization, and mapping.
- **Decision Making:** Algorithms and logic that enable the robot to make decisions based on sensor inputs and predefined rules or objectives, including path planning, obstacle avoidance, and task prioritization.

- **Communication:** Protocols and methods for communication between the robot and external devices or systems, such as remote control interfaces, network communication, or communication with other robots in a multi-robot system.
- **Error Handling and Recovery:** Strategies for detecting errors or failures in the robot's operation and implementing appropriate responses or recovery procedures to ensure safe and reliable performance.
- **Human-Robot Interaction:** Interfaces and behaviors that enable interaction between the robot and humans, including speech recognition, gesture recognition, and user interfaces for commanding or monitoring the robot's activities.
- **Integration with Higher-Level Systems:** Integration of robot programming with higher-level systems or software frameworks for task planning, scheduling, monitoring, and coordination in complex robotic systems.

5) Define Arduino

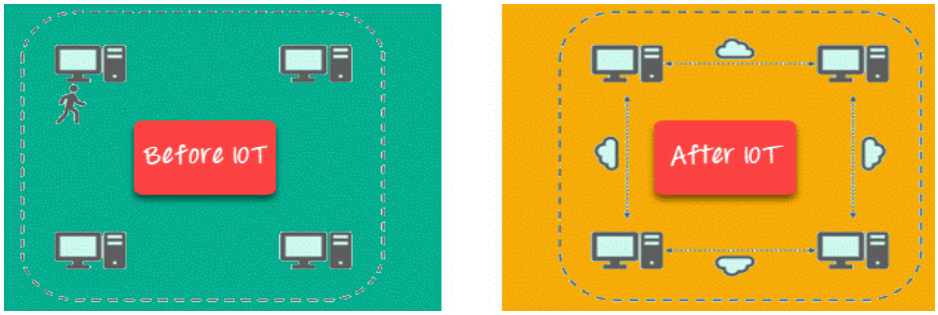
Arduino is a free electronics platform having easy to use hardware and software. It has a microcontroller capable of reading input from sensors to control the motors programmatically.

6) List mostly used sensors types in Robot programming.

Mostly used sensor types in Robot Programming are:

- Smoke sensor
- Temperature sensors
- Pressure sensor
- Motion detection sensors
- Gas sensor
- Proximity sensor
- IR sensors

7) What are the advantages of IoT in Robotics?



Key benefits of IoT technology are as follows:

- **Technical Optimization:** IoT technology helps a lot in improving techniques and making them better. For example, with IoT, a manufacturer is able to collect data from various car sensors. The manufacturer analyses them to improve its design and make them more efficient.
- **Improved Data Collection:** Traditional data collection has its limitations and its design for passive use. IoT facilitates immediate action on data.
- **Reduced Waste:** IoT offers real-time information leading to effective decision making & management of resources. For example, if a manufacturer finds an issue in multiple car engines, he can track the manufacturing plan of those engines and solves this issue with the manufacturing belt.
- **Improved Customer Engagement:** IoT allows you to improve customer experience by detecting problems and improving the process.

8) What is Bluegiga APX4 protocol?

The Bluegiga APX4 is a solution that supports both the WiFi and BLE platform, and it is based on a 450MHz ARM9 processor.

9) What is Pulse Width Modulation?

PWM or Pulse Width Modulation is a variation of how much time the signal is high in an analog fashion. The signal can be high or low and the user can even change the proportion of the time.

10) Mention applications of PWM in IoT

Applications of PWM in IoT are controlling the speed of DC motor, controlling the direction of a servo moto, Dimming LED, etc.

11) List available wireless communications boards available in Raspberry Pi?

Wireless communications boards available in Raspberry Pi are 1) WiFi and 2) BLE/Bluetooth.

12) What are the functions used to read analog and digital data from a sensor in Arduino?

Functions used to read analog and digital data from a sensor in Arduino are: digitalRead() and digitalWrite().

13) What is Bluetooth Low Energy?

Bluetooth Low Energy is a wireless PAN (Personal Area Network) technology. It uses less power to transmit long-distance over a short distance.

14) Define Micro Python

MicroPython is a Python implementation, which includes a small subset of its standard library. It can be optimized to run on the ModeMCU microcontroller.

15) List available models in Raspberry Pi

Models of Raspberry Pi are:

- Raspberry Pi 1 Model B
- Raspberry Pi 1 Model B+
- Raspberry Pi 1 Model A
- Raspberry Pi Zero
- Raspberry Pi 3 Model B
- Raspberry Pi 1model A+
- Raspberry Pi Zero W
- Raspberry Pi 2

16) Mention some of the commonly used water sensors

The commonly used water sensors are:

- Turbidity sensor.
- Total organic carbon sensor.
- pH sensor.
- Conductivity sensor.

17) Differentiate between Arduino and Raspberry pi

Arduino	Raspberry pi
Arduino is an open, programmable USB	It can execute one program at a time.

microcontroller.	
Raspberry pi is a credit card size computer.	Users can run more than one program at a time.

18) What are mostly used IoT protocols?

The mostly used IoT protocols are:

- XMPP
- AMQP
- Very Simple Control Protocol (VSCP)
- Data Distribution Service (DDS)
- MQTT protocol
- WiFi
- Simple Text Oriented Messaging Protocol(STOMP)
- Zigbee

19) What is a library in Arduino?

Arduino library is a collection of code that is already written for controlling module or sensor.

20) Mention some of the wearable Arduino boards

Wearable Arduino boards are:

- Lilypad Arduino main board.
- Lilypad Arduino simple.
- Lilypad Arduino simple snap.
- Lilypad Arduino USB.

21) What is the main difference between floating CPU and fixed-point CPU?

Floating CPU can take floating value directly, whereas fixed CPU is converted to integer format. Thereby, it leads to the loss of some resolution.

22) Define GPIO.

GPIO is a programmable pin that can be used to control input or output pins programmatically.

23) Explain Android things.

Android things are an Android-based OS that is built for embedded devices.

24) What is the aim of airflow sensors?

The main aim of airflow sensors is to measure the air level in the soil. This sensor enables one to measure it dynamically, from one location, or multiple locations of the garden.

25) Why use the scheduler in RTOS?

Scheduler in RTOS is used for switching one task to another.

26) Mention real-time usage of Raspberry pi.

- Home automation
- Portable web server
- manipulating the robots
- Internet radio

27) What is WSN?

The full form of WSN is Wireless Sensor Network. It is a network of notes, design to observe and to study physical parameters of the application.

28) What is Zigbee?

Zigbee is the same like Bluetooth. It used in a complex system for low power operation, robustness, and high security.

29) How to install a new library in Arduino?

A new library in Arduino can be installed by selecting the library from the sketch option in Toolbar.

30) What are the operating systems supported by Pi?

Operating systems supported by Pi are:

- Raspbian
- Open ELEC (Open Embedded Linux Entertainment center)
- RISC OS
- Lakka
- OSMC (Open Source Media Centre)
- Windows IoT Core

31) How to reduce the size of the sketch?

Reducing the size of the sketch is can be reduced by removing unwanted libraries from the code and make code short and simple.

32) What is the difference between M2M and IoT?

The difference between M2M and IoT is:

M2M	IoT
Communication is done within embedded software at the client site.	Communication is done for grand-scale projects.
It uses isolated systems of devices having the same standards.	It uses integrated devices, applications, and data across varying standards.
M2M offers limited scalability options.	IoT is inherently more scalable.
A cellular network or wired network is used for device connectivity.	It uses an active Internet connection for device connectivity.
Machines can communicate with one machine at a time.	Many machines can communicate with each other over the Internet.

33) How to program Arduino?

Programmers can use the Arduino IDE in order to write an Arduino program. Developers can also use Node.js Johny five-module in order to control Arduino.

34) What are IoT testing tools?

IoT testing tools can be divided into hardware and software:

- **IoT testing software:** Tcpdump and Wireshark.
- **Hardware for IoT testing:** JTAG Dongle, Digital Storage Oscilloscope, and Software Defined Radio.

35) How to store the high-volume file into Arduino?

A specification called Gridfs can be used for storing high volume file into Arduino.

36) What is Shodan?

Shodan is an IOT testing tool that can be used to discover which of your devices are connected to the Internet. It allows you to keep track of all the computers which are directly accessible from the Internet.

37) Mention some examples of MEMS sensor.

- MPU6050- Gyroscope
- ADXL345
- piezoelectric sensor
- Accelerometer

38) What is sharding?

Sharding is a method to split data into collections and stored in machines.

39) What is Thingful?

Thingful is a search engine for the Internet of Things. It allows secure interoperability between millions of IoT objects via the Internet. This IOT testing tool also controls how data is used and empowers to take more decisive and valuable decisions.

References:

1. **"Robotics: Modelling, Planning and Control" by Bruno Siciliano and Lorenzo Sciavicco**: This comprehensive textbook covers a wide range of topics in robotics, including kinematics, dynamics, motion planning, and control, making it an excellent reference for understanding the theoretical aspects of robotic programming.

2. **"Programming Robots with ROS: A Practical Introduction to the Robot Operating System" by Morgan Quigley, Brian Gerkey, and William D. Smart**: This book provides a practical introduction to the Robot Operating System (ROS), a widely used framework for robot programming, including tutorials and examples for building and programming robots using ROS.

3. **"Probabilistic Robotics" by Sebastian Thrun, Wolfram Burgard, and Dieter Fox**: This book covers probabilistic methods and algorithms for robot perception, localization, mapping, and navigation, providing a foundation for understanding how robots perceive and interact with their environment.

4. **Online Courses and Tutorials**: Websites like Coursera, edX, and Udacity offer online courses and tutorials on robotics and robot programming, covering topics such as kinematics, control theory, sensor integration, and ROS programming.

5. **Research Papers and Journals**: Academic journals such as IEEE Transactions on Robotics and Automation, International Journal of Robotics Research, and conferences like IEEE International Conference on Robotics and Automation (ICRA) and Robotics: Science and Systems (RSS) publish research papers on various aspects of robotics and robot programming, providing insights into the latest developments and techniques in the field.

6. **Open-Source Robotics Software and Documentation**: Open-source robotics software libraries like ROS, Gazebo, and RobotPy provide extensive documentation, tutorials, and examples for robot programming, making them valuable references for learning

and implementing robotic systems.