MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF MECHATRONICS ENGINEERING

ROBOTICS LAB -I MANUAL (MTE 2162)

3rd Semester B.Tech (Mechatronics)

NAME: ______________________________

REG NO: ____________________________

ROLL NO: ___________________________

VISION OF THE MECHATRONICS ENGINEERING DEPARTMENT

Excellence in Mechatronics Education through Innovation and Team Work.

MISSION OF THE MECHATRONICS ENGINEERING DEPARTMENT

Educate students professionally to face societal challenges by providing a healthy learning environment grounded well in the principles of Mechatronics engineering, promoting creativity, and nurturing teamwork.

NBA PROGRAM EDUCATIONAL OUTCOMES OF THE MECHATRONICS ENGINEERING DEPARTMENT (PEOs)

The graduates**:**

PEO1: Are expected to apply analytical skills and modelling methodologies to recognize, analyze, synthesize and implement operational solutions to engineering problems, product design and development, and manufacturing.

PEO2: Will be able to work in national and international companies as engineers who can contribute to research and development and solve technical problems by taking an initiative to develop and execute projects and collaborate with others in a team.

PEO3: Shall be capable of pursuing higher education in globally reputed universities by conducting original research in related disciplines or interdisciplinary topics, ultimately contributing to the scientific community with novel research findings.

PEO4: Are envisioned to become technology leaders by starting high – tech companies based on social demands and national needs.

PEO5: Shall develop flexibility to unlearn and relearn by being in pursuit of research and development, evolving technologies and changing societal needs thus keeping themselves professionally relevant.

NBA PROGRAM OUTCOMES (PO):

The POs are exemplars of the attributes expected of a graduate of an accredited programme:

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

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

PO3- Design solutions for complex engineering problems and design system components or processes that meet t h e specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4- 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- Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6- 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- Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

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

PO9- Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10- 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- Demonstrate knowledge and understanding of t h e 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- 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.

NBA PROGRAM SPECIFIC OUTCOMES OF THE MECHATRONICS ENGINEERING DEPARTMENT (PSOS)

At the end of the course the student will be able to:

PSO1: Apply the knowledge of sensors, drives, actuators, controls, mechanical design and modern software tools to integrate a system for performing specified tasks.

PSO2: Articulate designs, modelling, analysis, and testing of Mechatronics products, systems and controllers using appropriate technology and software tools.

LIST OF EXPERIMENTS:

Sl. No.	TITLE OF EXPERIMENT	DATE	PAGE NO.	FACULTY SIGN	MARKS
1.	A. Introduction to Raspberry Pi programming. B. Read sensor data and glow LEDs using Raspberry Pi.				
2.	Control of servo motor actuators using Raspberry Pi.				
3.	Color detection using opencv.				
4.	Shape detection using opencv				
5.	Biscuit Quality detection using opencv				
6.	Robot Studio: Using the library tool trace a square. IRB (Online Programming): Jogging and tracing a square.				
7.	Robot Studio: Create a custom tool and trace a circle with auto path. IRB (Online Programming): Pick and place a single object/ multiple objects.				
8.	Universal Robots UR5 collaborative robot: Introduction, defining tool center point, implementation of pick and place operation				
9.	Universal Robots UR5 collaborative robot: Implementation of destacking and stacking operation				
10	Read DH-parameters and obtain the forward transformation matrix (Using Python)				
11.	Mini Project using Raspberry Pi/ Arduino/ OpenCV/ Robot Studio/Cobot				

List of Components available for Mini-project

In charge: Archana			
Sl. No	Components available for mini project	Number	Specifications
1.	Small Metal Chassis	5	
2.	Motors (300 RPM)	6	
3.	3D printing material (PLA)	2 bundles	
4.	BLDC motor	10	
5.	ESC	10 sets	
6.	Propellers	5 sets	
7.	Arduino NANO	02	
8.	Arduino GEMMA	02	
9.	Raspberry pi 4	02	4GB RAM
10	Raspberry pi 3	03	3GB RAM
11	Arduino UNO	06	
12	Arduino YUN	02	
13	Arduino Mega	04	
14	Arduino Due	04	
15	Bread board power supply (YW Robot)	04	
16	Arduino Mini	05	
17	LED	1 set	Red , Blue, Green, Yellow
18	Flow Sensor	04	
19	IR Sensor	10	
20	Ultrasonic Sensor	10	
21	Motor driver	06	L293
22	Accelerometer	05	
23	Fire sensor	04	
24	Voltage Regulator	10	LM328
25	LDR sensor	10	
26	Encoder	05	
27	FTD1232 USB to TTL serial connector adapter module	02	

28	Capacitive touch digital keypad	10	
29	LDR	10	
30	Relay 8 channel	03	
21	Relay 4 channel	05	
32	Relay 1 channel	01	
33	LCD	10	14*2
34	Stepper motor	05	25kg
34	Servo motor	05	
36	Solder gun	05	
37	Lead	05	
38	Flux	03	
39	Pulleys and gears	1 set	
40	Multimeter	02	
41	Wireless Nano Usb connector	01	500mbps
42	IMU breakout	12	
43	Single strand and Multi strand wires	1 bundle	
44	Wi-Fi shield Uno and Mega Compatible	02	
45	Ethernet shield for Arduino	01	
46	Breadboard	14	
47	Beagle bone	02	
48	USB adapter for Raspberry pi	03	
49	Wire stripper	05	
50	Drill machine with drill bit	01	
51	Screw driver set	02	
52	Allen keys	01 set	
53	Jumper wires	01 set	Male to female Male to male Female to female
54	Power connector to Arduino	10	
55	Crocodile clips	20	
56	Nuts and Screws and Washers	1 set	
57	Resistor	1 bunch	

58	Long USB connector	03	
59	ARM controller	04	
60	Lego connecting cable	04	
61	Scissor	01	
62	Potentiometer	05	
63	Power connector	04	
64	Lego kit	04	
65	Balance charger	04	12V
66	DC Regulated power supply	02	3005
67	DC Regulated power supply	03	302D
68	LED Acid battery	04	12v and 1.3AH
69	Real sense depth camera	03	
70	Logitech web camera	02	
71	RP Lidar	05	
72	Clear path Turtlebot 2	02	
73	My Rio kit	01	
74	Elvis kit	01	

Safety Instructions:

A robot is heavy and extremely powerful regardless of its speed. A pause or long stop in movement can be followed by a fast hazardous movement. Even if a pattern of movement is predicted, a change in operation can be triggered by an external signal resulting in an unexpected movement. Therefore, it is important that all safety regulations are followed when entering safeguarded space.

Safety regulations

Before beginning work with the robot, make sure you are familiar with the safety regulations

1. *Described in ABB Operating manual - IRC5 with Flex Pendant.*
2. *Robotics Lab safety instructions*
3. *Dos and Dont's*

Robotics Lab Safety Instructions:

1. *Don't enter into the fenced area of the robot while the robot is idle or working.*
2. *Wear safety shoe while working in the Laboratory.*
3. *Don't try to operate the IRB Robot without prior authorization of the lab in charge/lab instructor.*
4. *Don't operate /program the Robot at the full speed mode.*
5. *No unauthorized experiments/programs should be performed.*
6. *Do not eat or drink in the laboratory.*
7. *In case of any accident involving the IRB robot press emergency stop switch located at the fence door, control panel or teach pendant.*
8. *Before beginning to work in the laboratory you should be familiar with the standard operating procedure of each and every equipment/component you may be using.*
9. *In case of an accident or any unexpected event like short circuit, burning of components, sparks in or around the workplace, immediately inform to the faculty or lab instructor.*
10. *Before switching ON the Robot ensure that no one is inside the fenced area and ensure that the compressor pressure is at 6bar.*

WORK INSTRUCTIONS

1. *Enter your details in the log register.*
2. *Listen to the instructions given by the faculty.*

3. *Identify the components and interfaces required for the approved experiment.*
4. *Write the indent for collecting the components from lab technician.*
5. *Rig the circuit/program the Robot/build the robot as per the prior instructions given by the faculty.*
6. *Before switching ON the IRB robot inform the lab faculty.*
7. *After conducting the experiment switch OFF the equipment/Shutdown the PC and handover the components borrowed to the lab technician.*

1. INTRODUCTION TO RASPBERRY PI (RPi)

A. Introduction to Raspberry Pi programming.

Aim: To learn the functionalities of Raspberry Pi

Theory: Raspberry Pi (RPi) is a credit card sized computer. It has ARM based Broadcom Processor SoC (System on Chip) along with on-chip GPU. Unlike Arduino board, one can connect all the peripherals like monitor, keyboard, mouse, camera etc., and use RPi board like a full-fledged computer. It also provides access to on-chip hardware. Raspbian OS is the official open source operating system to be installed in RPi board.

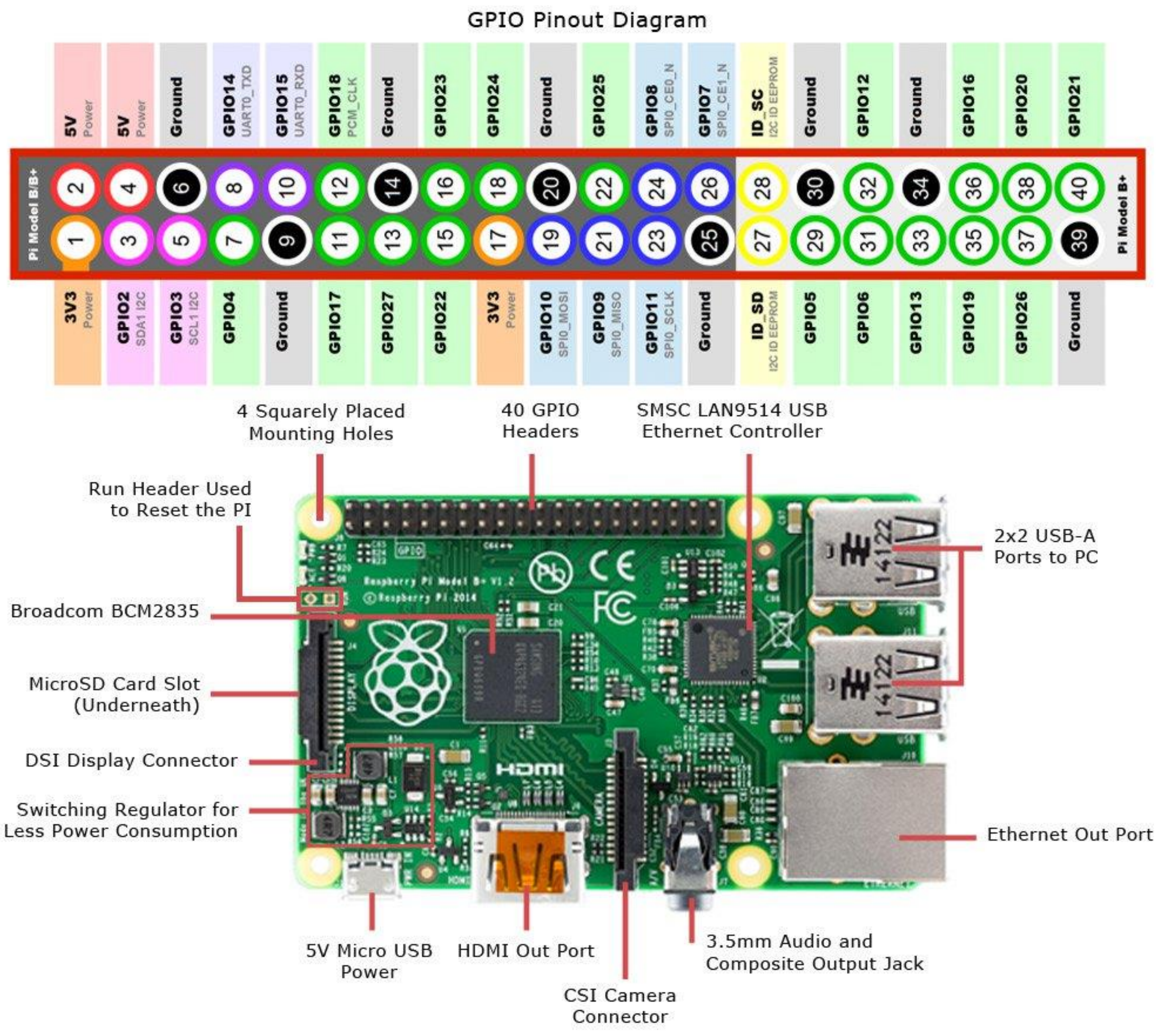


Figure 4.1: Raspberry Pi board and pinout diagram.

Some of the features in RPi board as shown in fig 4.1 is listed below.

HDMI (High-Definition Multimedia Interface): It is used for transmitting uncompressed video or digital audio data to the Computer Monitor, Digital TV, etc. Generally, this HDMI port helps to connect Raspberry Pi to the Digital television.

CSI Camera Interface: CSI (Camera Serial Interface) interface provides a connection in between Broadcom Processor and Pi camera. This interface provides electrical connections between two devices.

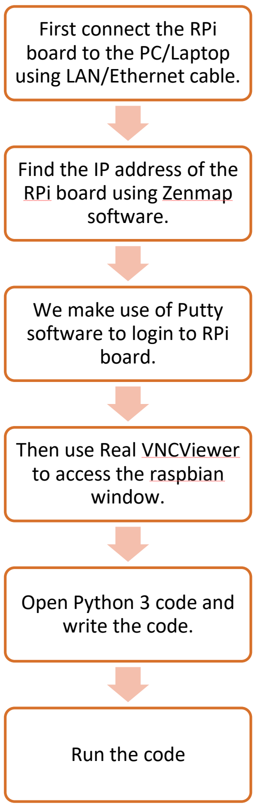
DSI Display Interface: DSI (Display Serial Interface) Display Interface is used for connecting LCD to the Raspberry Pi using 15-pin ribbon cable. DSI provides fast High-resolution display interface specifically used for sending video data directly from GPU to the LCD display.

Composite Video and Audio Output: The composite Video and Audio output port carries video along with audio signal to the Audio/Video systems.

Power LED: It is a RED colored LED which is used for Power indication. This LED will turn ON when Power is connected to the Raspberry Pi. It is connected to 5V directly and will start blinking whenever the supply voltage drops below 4.63V.

ACT PWR: ACT PWR is Green LED which shows the SD card activity.

Procedure for connecting RPi board to computer (Annexure 1):



First connect the RPi board to the PC/Laptop using LAN/Ethernet cable.

Find the IP address of the RPi board using Zenmap software.

We make use of Putty software to login to RPi board.

Then use Real VNCViewer to access the raspbian window.

Open Python 3 code and write the code.

Run the code

B. Reading and writing digital signal.

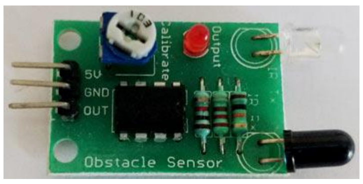


Figure 4.2: A typical IR sensor.

Example Code:

```
import RPi.GPIO as IO          # Importing GPIO file from library and renaming it as IO.
from time import sleep         # Import sleep function from time library
IO.setmode (IO.BCM)           #Refering the GPIO pins on the RPi board either by pin number
on
                              #board or by their function.
IO.setup(5,IO.OUT)            #GPIO 5 -> Red LED as output
IO.setup(6,IO.OUT)            #GPIO 6 -> Green LED as output
```

```
while True:
        IO.output(5,True)      # led ON
        IO.output(6,False)     # led OFF
        print(" 5 ON, 6 OFF")
        sleep(2)
        IO.output(5,False)      # led OFF
        IO.output(6,True)      # led ON
        print(" 5 OFF, 6 ON")
        sleep(2)
```

Exercise:

- Interface an IR sensor to the RPi board. When the object is detected glow GREEN LED else glow RED LED. (For useful syntax refer annexure)

2. CONTROL A SERVO MOTOR USING RASPBERRY PI (RPi)

Aim: To control the direction of rotation of a basic servo motor using raspberry pi.

Components required: Servo motor, LEDs, connecting wires, raspberry pi board.

Theory:

Servos can be controlled by sending an electrical pulse of variable width (PWM) to the control input. Generally the time period of the pulse is 20ms and 1.5ms ON time rotates the servo to 90 degree.

Circuit Diagram:

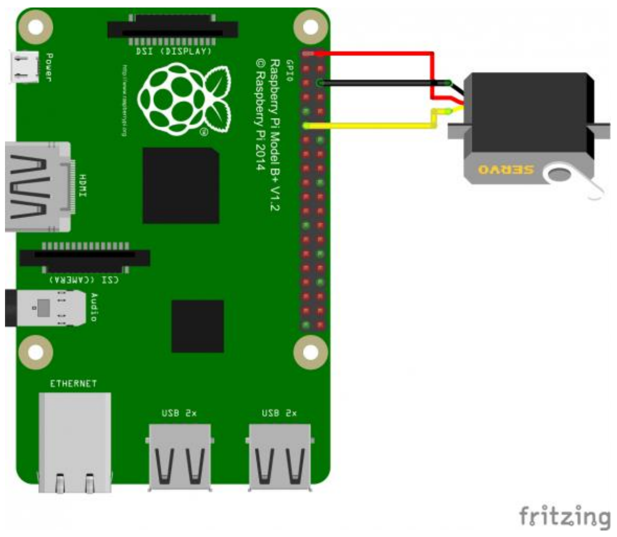


Figure 5.1: Hardware connection for servo motor control using RPi.

Servo motor has three connections to be made. Two connections for power supply and ground. Third wire is for control signal.

RPi code:

- Import the GPIO module
- Import sleep from time library
- Choose the RPi board configuration
- Configure the output pin for servo control signal
- Set the PWM frequency (Time period of the PWM)
- Start the pwm signal with 0 (zero) duty cycle
- Next set the angle servo should rotate

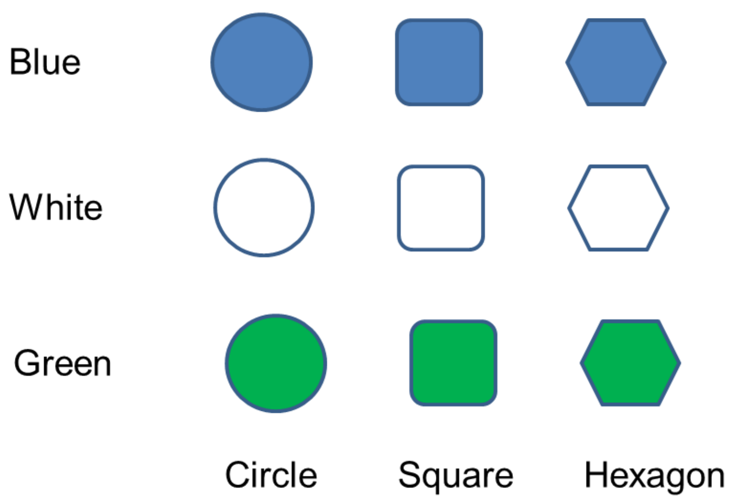
Exercise:

- Write a RPi code to detect an object using IR sensor and accordingly rotate the servo to 90 degrees. If the object is not detected rotate the servo to 0 degrees.
- Write a RPi code to detect objects moving on a conveyor, on counting 5 objects rotate the servo to 90 degrees.

3- COLOR DETECTION USING OPENCV

Aim: Identification and classification of the objects based on color based on HSV values

Images of objects to be classified based on shape



Procedure

Install Anaconda for Windows: Python 3.8

Install opencv:

Open anaconda prompt and type: pip3 install opencv-python

Install Jupyter notebook:

Open anaconda prompt and type: pip3 install jupyter

Open Jupyter notebook:

Open anaconda prompt and type: jupyter notebook

Type the following code in Jupyter Notebook to identify color green:

```
import cv2
import numpy as np

image=cv2.imread('D:/2021/Laboratories/Robotics lab/opencv/0001.bmp')
cv2.imshow("image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()


scale_percent = 60
```

```
width = int(image.shape[1] * scale_percent / 100)
height = int(image.shape[0] * scale_percent / 100)
dim = (width, height)

# resize image
img = cv2.resize(image, dim, interpolation = cv2.INTER_AREA)
cv2.imshow("image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()


refPt=[(width//2-50, height//2-50),(width//2+50, height//2+50)]
roi = img[refPt[0][1]:refPt[1][1], refPt[0][0]:refPt[1][0]]
cv2.imshow("ROI", roi)
cv2.waitKey(0)
cv2.destroyAllWindows()

import numpy as np
hsv=cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
havg= np.average(hsv, axis=0)
h_avg = np.average(havg, axis=0)
h_avg

greenLower=(90,100,75)
greenUpper=(100,180,200)


if (h_avg[0]>np.array(greenLower[0])) and  (h_avg[0]<np.array(greenUpper[0])) and
(h_avg[1]>np.array(greenLower[1])) and  (h_avg[1]<np.array(greenUpper[1])) and
(h_avg[2]>np.array(greenLower[2])) and  (h_avg[2]<np.array(greenUpper[2])):
    print("Color is green")
else:
    print("Color is not green")
```

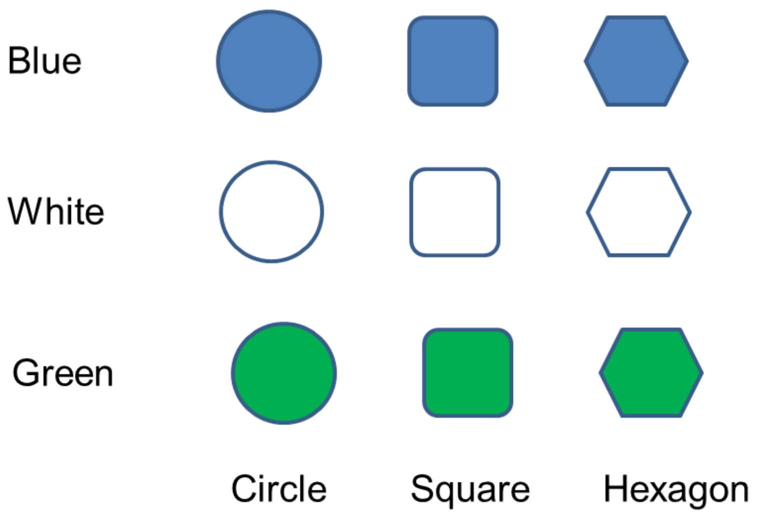
EXERCISE:

Modify the code to identify colors white, blue and green on the basis of HSV values

4- SHAPE DETECTION USING OPENCV

Aim: Identification and classification of the objects based on color using suitable algorithms.

Images of objects to be classified based on color



Procedure:

- Import the images in opencv
- Blur the image using Gaussian Blur
- Convert the image from color to gray
- Perform edge detection based on thresholding
- Perform dilation to get a single contour
- Find the contours in the image
- For each of the contours determined in the image, the approxPolyDP() function is applied to determine the shape of the polygons present in the image.
- Then the determined shape of the contours is drawn on the image using drawContours() function and displayed as the output of the program.
- The shape is also printed on the image as a text.

Program:

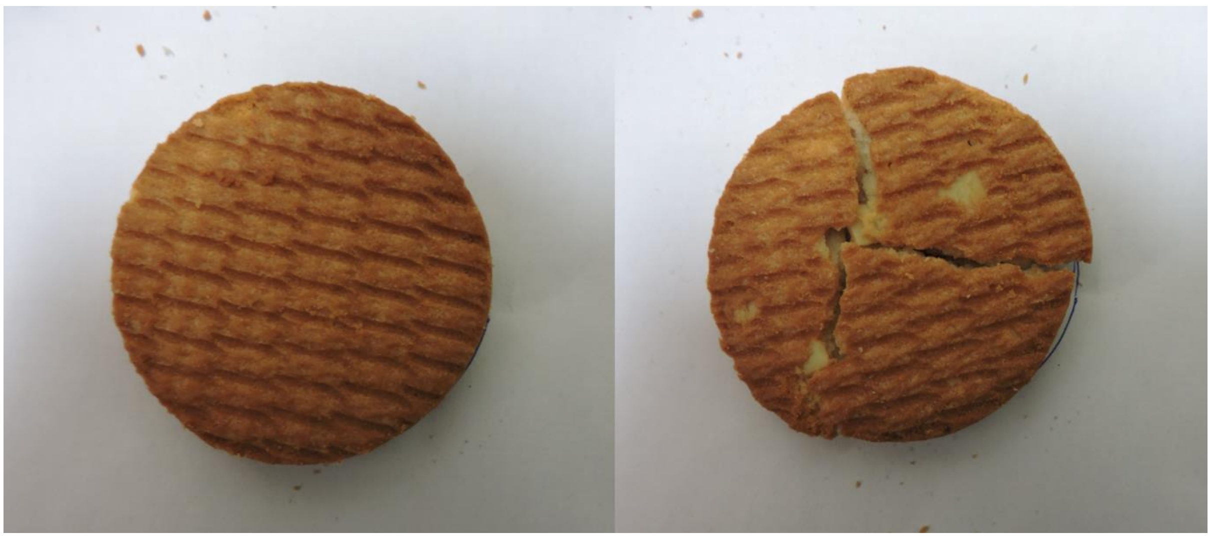
```
import numpy as np
import cv2
img = cv2.imread('D:/2021/Laboratories/Robotics lab/opencv/New images/7.jpg')

Blur=cv2.GaussianBlur(img, (1,1), 0)
cv2.imshow("Blur",Blur)
cv2.waitKey(0)
Gray=cv2.cvtColor(Blur,cv2.COLOR_BGR2GRAY)
cv2.imshow("Gray",Gray)
cv2.waitKey(0)
ret,thresh=cv2.threshold(Gray,… ,…,1)
cv2.imshow('thresh', thresh)
cv2.waitKey(0)
kernel = np.ones((…,…))
Dilate=cv2.dilate(thresh, kernel, iterations=1)
cv2.imshow('Dilate', Dilate)
cv2.waitKey(0)

contours , hierarchy = cv2.findContours(Dilate.copy(), cv2.RETR_TREE,
          cv2.CHAIN_APPROX_NONE)
for contour in contours:
   approx = cv2.approxPolyDP(contour, 0.01* cv2.arcLength(contour, True), True)
   cv2.drawContours(img, [approx], 0, (0, 0, 0), 5)
   x = approx.ravel()[0]
   y = approx.ravel()[1] - 5
   if len(approx) == 3:
      cv2.putText( img, "Triangle", (x, y), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0) )
   elif len(approx) == …:
      x, y , w, h = cv2.boundingRect(approx)
      aspectRatio = float(w)/h
      print(aspectRatio)
      if aspectRatio >= …. and aspectRatio < …:
         cv2.putText(img, "square", (x, y), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0))
      else:
         cv2.putText(img, "rectangle", (x, y), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0))

   elif len(approx) == … :
      cv2.putText(img, "pentagon", (x, y), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0))
   else:
      cv2.putText(img, "circle", (x, y), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0))

cv2.imshow('shapes', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Exercise:

Determine the values marked as '…' to detect all the given shapes successfully.

5. BISCUIT QUALITY DETECTION USING OPENCV

Aim: Determination of biscuit quality from images



Theory:

Industries would require several quality inspection applications. One such application has been demonstrated for food industries. The quality of the biscuits has to be checked and classified as broken or not broken.

Procedure:

- Open Jupyter Notebook
- Import required libraries and read the image.
- Convert the color image to HSV image- to identify the biscuit only.
- Convert the HSV to a binary image by setting up H, S and V lower and upper thresholds
- Dilate the image to remove small imperfections
- Find the contours
- Compare the no. of contours, contour perimeter and area to detect broken or good quality biscuits.

Program:

```
import cv2
import numpy as np

image=cv2.imread('D:/2021/Laboratories/Robotics lab/opencv/Biscuit/3.jpg')
cv2.imshow("image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()

def mouse(event,x,y,flags,param):
        if event==cv2.EVENT_LBUTTONDOWN:
                h=hsv[y,x,0]
                s=hsv[y,x,1]
                v=hsv[y,x,2]
                print("H:",h)
                print("S:",s)
                print("V:",v)

cv2.namedWindow('mouse')
cv2.setMouseCallback('mouse',mouse)
hsv=cv2.cvtColor(image,cv2.COLOR_BGR2HSV)
cv2.imshow("mouse", hsv)
cv2.waitKey(0)
cv2.destroyAllWindows()

redLower=(…,…,…)
redUpper=(…,…,…)

mask=cv2.inRange(hsv,redLower,redUpper)
cv2.imshow("mask",mask)
cv2.waitKey(0)
cv2.destroyAllWindows()

kernel=np.ones((…,…),np.uint8)
mask=cv2.dilate(mask,kernel,iterations=2)
contours,heirarchy=cv2.findContours(mask,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX
_NONE)
cv2.imshow("dilate",mask)
cv2.waitKey(0)
cv2.destroyAllWindows()
print("Number of contours found="+str(len(contours)))

for c in contours:
        area = cv2.contourArea(c)
        perimeter = cv2.arcLength(c, True)
        ((x,y),radius)= cv2.minEnclosingCircle(c)
        if (area>…):
                cv2.drawContours(image,[c],-1,(255,0,0),5)
                print("Area.{},Perimeter.{}".format(area,perimeter))
        print("number of contours:{}".format(len(contours)))
        cv2.imshow("RGB", image)
        if(perimeter>…):
            string="broken"
        else:
            string="good quality"
cv2.waitKey(0)
cv2.destroyAllWindows()


cv2.putText(image,string, (50,50), cv2.FONT_HERSHEY_COMPLEX,2,(0,0,255), 2)
cv2.imshow("contour",image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Exercise:

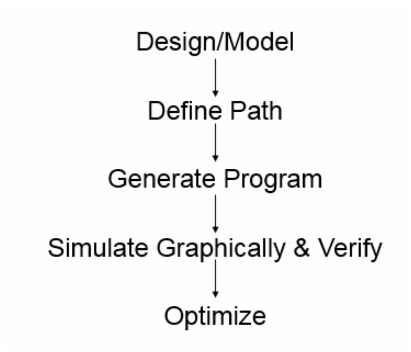
Determine the values marked as '…' to detect all the given shapes successfully.

6. INTRODUCTION TO ROBOT STUDIO AND IRB ONLINE PROGRAMMING

A. Using the library tool trace a square in Robot Studio.

Theory: An offline robot programming software package by ABB for programming all ABB robots that run on IRC and similar controllers. Robot studio is basically used to write robot programs without using the teach pendant/Flex pendant. Robot studio can be used to define a "Robot World" and the paths the robot takes in these worlds to complete its tasks. This is done though a graphical interface. The software then automatically generates the RAPID code in a file which can be later transferred to actual robot controller through a proper network like Ethernet. **Targets** (positions) and paths (sequences of move instructions to targets) are used when programming robot motions in Robot Studio. When you synchronize the Robot Studio station to the virtual controller, RAPID programs are created from the paths. A target is a coordinate that the robot shall reach. It contains the position of the target, defined in a work object coordinate system. **Orientation:** The orientation of the target, relative to the orientation of the work object. When the robot reaches the target, it will align the TCP's orientation with the targets orientation. **Configuration:** Configuration values that specify how the robot shall reach the target.

Procedure for using robot studio:



Procedure for Creating Targets:

1. In the Layout browser, select the workobject in which you want to create the target.

2. Click **Create Target** to bring up a dialog box.

3. Select the Reference coordinate system you want to use to position the target if you want to position the target.

4. In the Points box, click **Add New** and then click the desired position in the graphics window to set the position of the target. You can also enter the values in the Coordinates boxes and click **Add**.

5. Enter the Orientation for the target. A preliminary cross will be shown in the graphics window at the selected position. Adjust the position, if necessary. To create the target, click **Create**.

6. If you want to change the workobject for which the target is to be created, expand the **Create Target** dialog box by clicking the **More button**. In the Workobject list, select the workobject in which you want to create the target.

7. If you want to change the target name from the default name, expand the **Create Target** dialog box by clicking the **More button** and entering the new name in the Target name box.

8. Click Create. The target will appear in the browser and in the graphics window.

Targets on Edge creates targets and move instructions along the edges of the geometric

surface by selecting target points in the graphics window. Each point on a geometric

edge has certain properties that can be used to position robot targets relative to the edge.

Procedure for Creating Path

1. In the **Paths & Targets** browser, select the folder in which you want to create the path.

2. Click **Empty Path**.

3. To set the correct motion properties for the targets, select the active process in the **Change Active Process** box in the **Elements** toolbar.

Procedure to generate Autopath:

1. In the Home tab, click Path and select AutoPath. The AutoPath dialog box appears.

2. Select the edge or curve of the geometric object for which you want to create an autopath. The selection is listed as edges between <Start> and <End> in the graphic window.

3. Click **Remove** to delete the recently added edge from the graphic window.

4. Click **Create** to generate a new autopath.

5. Click **Close.**

B. IRB: Jogging and trace a square.

Theory :

Jogging is the process of moving the robot manually with the teach pendant for teaching the target points. RAPID is a high level programming language. It uses English words (like IF and FOR) to make it understandable for humans.

Procedure for Jogging

1. Go to the ABB Menu.
2. Click on Jogging tab.
3. Select Mechanical Unit as IRB Robot click ok
4. Select the motion mode as Axis 1-3, 4-6, / Linear/ Re-orient and click Ok
5. Enabling the guard stop move the joystick accordingly so as to bring the robot to the required target point.

Procedure for defining targets and there by trace a path:

1. Select manual mode using the key switch.
2. Click ABB Menu > Program editor > Tasks & Program Tab select ROB1 as task and click ok.
3. Click on modules>Click on File > Click on New Module > Click on yes > Name the new model & press OK > Click on created module >show module
4. Click on routine tab >file>new routine (type the name of new routine & press OK) > Click on show routine >add instructions and jog the robot to the required target point (Home position).
5. Click on add instruction > select MoveJ
6. Double Click on Robtarget(*) >New (This is the Home Position with address as ‘P*’, where * can be multiples of 10) > Select the tool as Pointer in the MoveJ instruction and Click OK.
7. Jog & bring end effector to the desired target point (Corner of the square) Click on add instruction >MoveL (a dialog box will appear, click on Below) Again Jog & bring end effecter to the desired point, (repeat above steps until all target points are thought).
8. To execute the program, click on Debug > PP to routine (PP- Program Pointer) > select OK.
9. Holding the guard switch press the play button on teach pendent.

7. Creating a Custom Tool in Robot Studio and Pick & Place objects using RAPID programming

Aim: A. To create a custom tool and trace a path.

B. Pick and place objects from table-1 and place it into the bin.

Theory:

Tools: One can model different types of tools such as single point and multipoint tools using the modelling tabs in robot studio and attach the same to the wrist of the Robot. There is an alternate option of importing the geometry (tool) from any modeling packages such as solid works ,Catia Pro E etc .

Workobject coordinate system: A workobject coordinate system is the coordinate system with reference to the position of a work piece.

The workobject consists of two frames: a user frame and an object frame. All programed positions will be related to the object frame, which is related to the user frame, which is related to the world coordinate system.

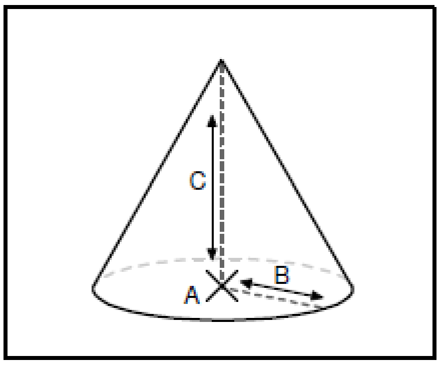
Procedure for creating a Pen or Pointer tool in Robot Studio:

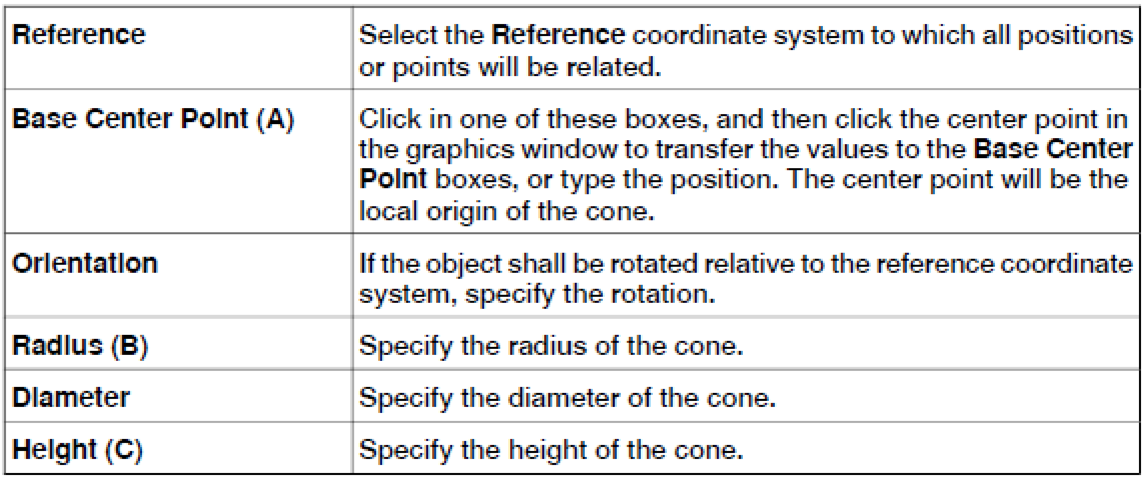
Creating a solid

1. Click **Solid** and then click **the type of solid** you want to create to bring up a dialog box. choose the **Create Cone** dialog box.

2. Enter requested values in the dialog box and click **Create.**

The create Cone dialog box



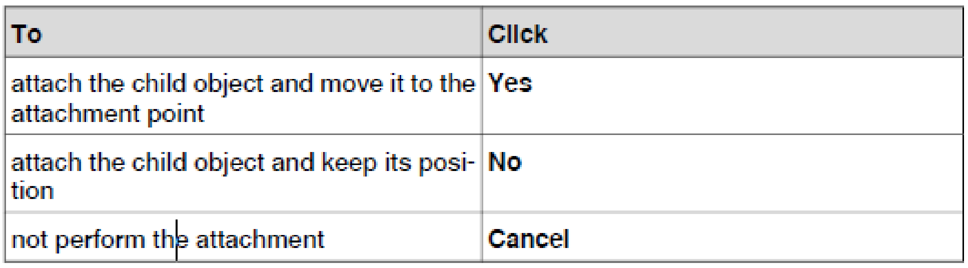


Reference	Select the **Reference** coordinate system to which all positions or points will be related.
Base Center Point (A)	Click in one of these boxes, and then click the center point in the graphics window to transfer the values to the **Base Center Point** boxes, or type the position. The center point will be the local origin of the cone.
Orientation	If the object shall be rotated relative to the reference coordinate system, specify the rotation.
Radius (B)	Specify the radius of the cone.
Diameter	Specify the diameter of the cone.
Height (C)	Specify the height of the cone.

3. Attaching a created pen tool to the Robot Wrist: In the Layout browser, right-click the child object, click Attach to and click the parent object in the list

4. Attaching an object/tool by drag and drop

- In the Layout browser, drag the child object to the parent object.
- In the displayed message, click the corresponding button



To	Click
attach the child object and move it to the attachment point	**Yes**
attach the child object and keep its position	**No**
not perform the attachment	**Cancel**

Procedure for creating a Workobject in Robot Studio:

1. In the Home tab (contains the controls required for building stations, creating systems, programming paths and placing items.) Click Workobject to bring up a dialog box. As shown in table 8.1

2. In the Misc Data group, enter the values for the new workobject.

3. In the User Frame group, do one of the following:

a. Set the position of the user frame by entering values for the Position x, y, z and the Rotation rx, ry, rz for the workobject by clicking in the Values box.

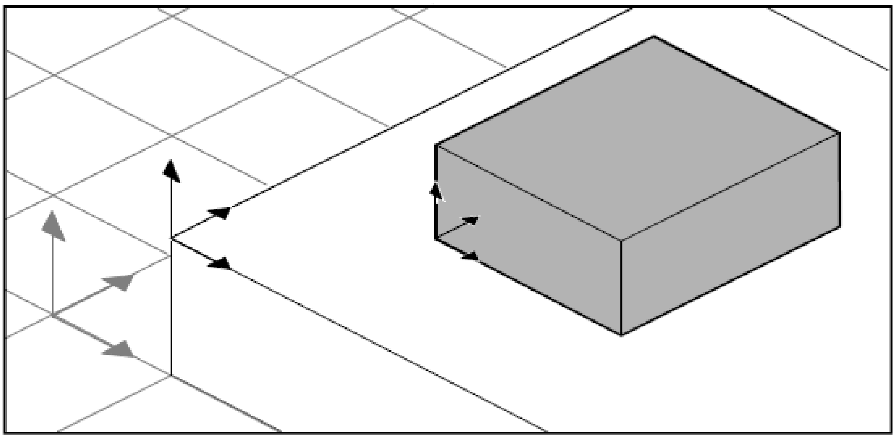
b. Select the user frame by using the Frame by points dialog box.

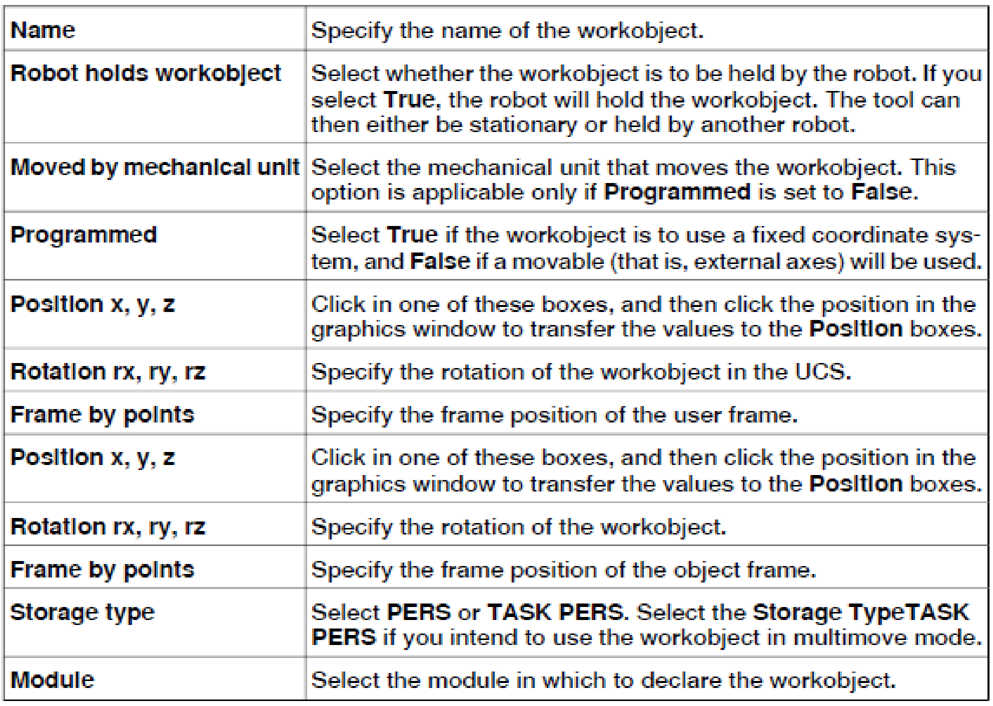
4. In the Object Frame group you can reposition the object frame relative to the user frame by doing any of the following:

a. a. Set the position of the object frame by selecting values for Position x,y, z by clicking in the Value box.
b. For the Rotation rx, ry, rz, select RPY (Euler XYX) or Quaternion, and enter the rotation values in the Values dialog box.
c. Select the object frame by using the Frame by points dialog box.

5. In the Sync Properties group, enter the values for the new workobject.

6. Click Create. The workobject will be created and displayed under the Targets node under the robot node in the Paths&Targets browser.





Name	Specify the name of the workobject.
Robot holds workobject	Select whether the workobject is to be held by the robot. If you select **True**, the robot will hold the workobject. The tool can then either be stationary or held by another robot.
Moved by mechanical unit	Select the mechanical unit that moves the workobject. This option is applicable only if **Programmed** is set to **False**.
Programmed	Select **True** if the workobject is to use a fixed coordinate system, and **False** if a movable (that is, external axes) will be used.
Position x, y, z	Click in one of these boxes, and then click the position in the graphics window to transfer the values to the **Position** boxes.
Rotation rx, ry, rz	Specify the rotation of the workobject in the UCS.
Frame by points	Specify the frame position of the user frame.
Position x, y, z	Click in one of these boxes, and then click the position in the graphics window to transfer the values to the **Position** boxes.
Rotation rx, ry, rz	Specify the rotation of the workobject.
Frame by points	Specify the frame position of the object frame.
Storage type	Select **PERS** or **TASK PERS**. Select the **Storage TypeTASK PERS** if you intend to use the workobject in multimove mode.
Module	Select the module in which to declare the workobject.

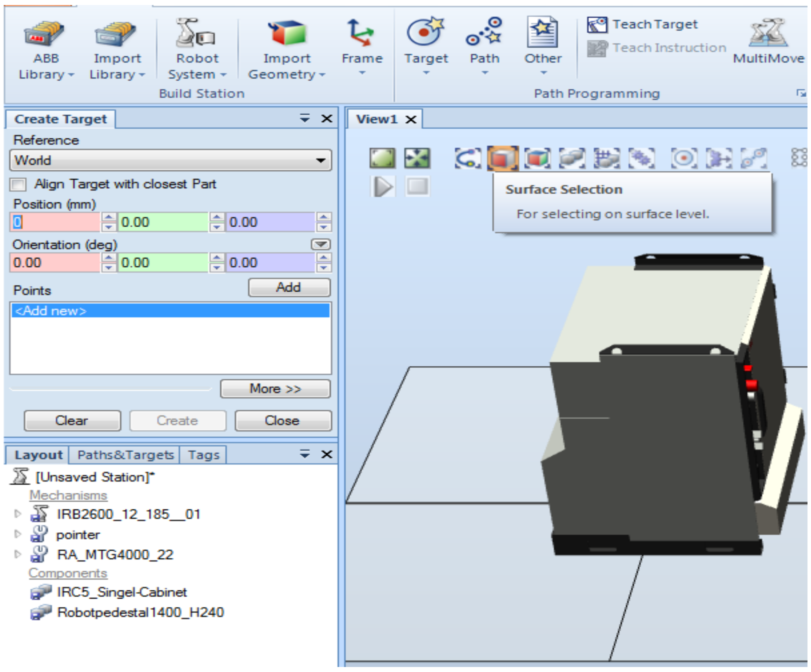
Table 8.1: Workobject dialog box

Procedure for Tracing a Square/Rectangle:

1. Click ABB library Tab>IRB 2600 or any other Robot
2. Click Import library Tab>equipment>IRC5 single cabinet (move and place the IRC5 as required using the move option in free hand Tab)
3. Click Import library Tab> user library>pointer (Crating pointer tool is discussed in previous experiment)
4. Click layout Tab>right click on pointer>click attach to IRB2600.
5. In settings Tab select task as IRB2600 and workobject as workobj0 and tool as pointer.
6. Click Import library>equipment>other>Robot pedestal. Move and place the pedestal in the layout suitably confirm whether the robot can reach all the points on the pedestal with ease.
7. Click on Robot system Tab>From layout>Finish(wait until the controller status turns green).
8. Click on Target Tab in path programming menu>surface selection>click on the required top surface of the pedestal as shown in the fig.2.1. also select the snap end option just beside the surface selection button.
9. Click on position dialog box > click on required corner/target points of the pedestal as shown in the fig 2.2.repeat this procedure for defining other corner/target points.
10. Click create on create target dialog box and click close.
11. In path and target Tab right click on Target_10>view tool at the target>pointer
12. Right click on Target_10>modify target>Click Rotate and enter the rotation angle and axis. Eg. Rotation about Y and angle as 180^0
13. Right click on Target_10>copy orientation
14. Holding control select all the other targets >Right click on Target_10>apply orientation.
15. In path programming tab click empty path
16. Select all the targets (Target_10,Target_20 etc.) Right click>add to path_10.
17. Right click on path>configuration>autoconfig. (check for Reach).
18. Right click on path>Insert Procedure call>path_10
19. Right click on path>rename and rename as main.
20. Click on synchronize tab in controller menu >Click on Synchronize with rapid.

21. Click play button in Simulation menu. The robot will trace the defined path square or rectangle in this case the edges of the pedestal.

22. For tracing a circular path create a solid cylinder from solid tab as discussed in experiment 1. Repeat steps 1- 14 and click path programming tab and select autopath.

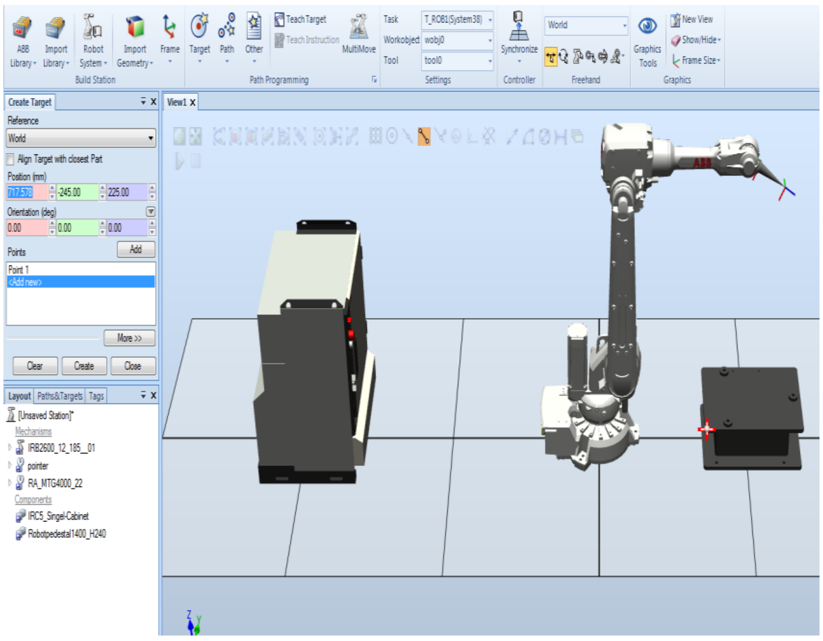


Reference: World
Align Target with closest Part
Position (mm): 0 | 0.00 | 0.00
Orientation (deg): 0.00 | 0.00 | 0.00
Points: <Add new>

Surface Selection
For selecting on surface level.

[Unsaved Station]*
Mechanisms
IRB2600_12_185__01
pointer
RA_MTG4000_22
Components
IRC5_Singel-Cabinet
Robotpedestal1400_H240

Fig 8.1: Snap shot showing the surface selection option.



Task: T_ROB1(System38)
Workobject: wobj0
Tool: tool0
World

Reference: World
Align Target with closest Part
Position (mm): 717.578 | -245.00 | 225.00
Orientation (deg): 0.00 | 0.00 | 0.00
Points: Point 1, <Add new>

[Unsaved Station]*
Mechanisms
IRB2600_12_185__01
pointer
RA_MTG4000_22
Components
IRC5_Singel-Cabinet
Robotpedestal1400_H240

Fig 8.2: Snap shot showing target selection option

B. **Problem Statement**: The robot will be clamped with an end effector which can grip and release an object that can be opened/closed with the digital output signal for a pneumatics direction control valve. Write a RAPID program to perform pick and place of multiple objects from table to bin.

Theory:

Signals are used for communication with external equipment that the robot cooperates with. Input signals are set by the external equipment and can be used in the RAPID program to initiate when to perform something with the robot. Output signals are set by the RAPID program and signals to the external equipment that they should do something.

Setting up signals:

Signals are configured in the system parameters for the robot system. It is possible to set customized names for the signals. They should not be declared in the RAPID program.

Digital input:

A digital input signal can have the values 0 or 1. The RAPID program can read its value but cannot set its value.

Example:

If the digital input signal di1 is 1 then the robot will move.

```
IF di1 = 1 THEN
MoveL p10, v1000, fine, tPen;
ENDIF
```

Digital output

A digital output signal can have the values 0 or 1. The RAPID program can set the value for a digital output signal, and thus affect external equipment. The value of a digital output signal is set with the instruction SET DO and can be reset using RESET DO.

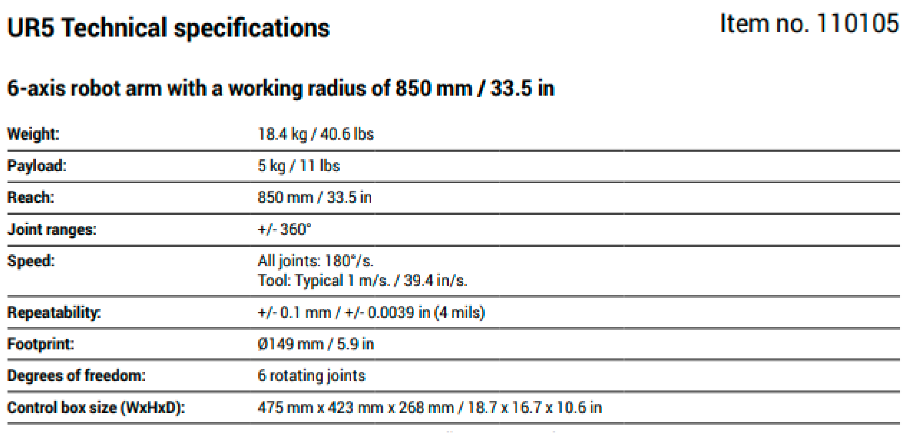
8. Introduction to UR5 COBOT- Implementation of Pick and Place operation

Aim: i. Understand basic operations: Jogging, setting up TCP, CoG, commands

ii. Implement Pick and Place Operation through programming with teachpendant

Theory:

The UR5e is a lightweight, adaptable collaborative industrial robot that tackles medium-duty applications with ultimate flexibility. Specifications of the COBOT are listed in Fig.8.1.



UR5 Technical specifications Item no. 110105

6-axis robot arm with a working radius of 850 mm / 33.5 in

Weight:	18.4 kg / 40.6 lbs
Payload:	5 kg / 11 lbs
Reach:	850 mm / 33.5 in
Joint ranges:	+/- 360°
Speed:	All joints: 180°/s. Tool: Typical 1 m/s. / 39.4 in/s.
Repeatability:	+/- 0.1 mm / +/- 0.0039 in (4 mils)
Footprint:	Ø149 mm / 5.9 in
Degrees of freedom:	6 rotating joints
Control box size (WxHxD):	475 mm x 423 mm x 268 mm / 18.7 x 16.7 x 10.6 in

Fig. 8.1: Specifications of UR5 COBOT

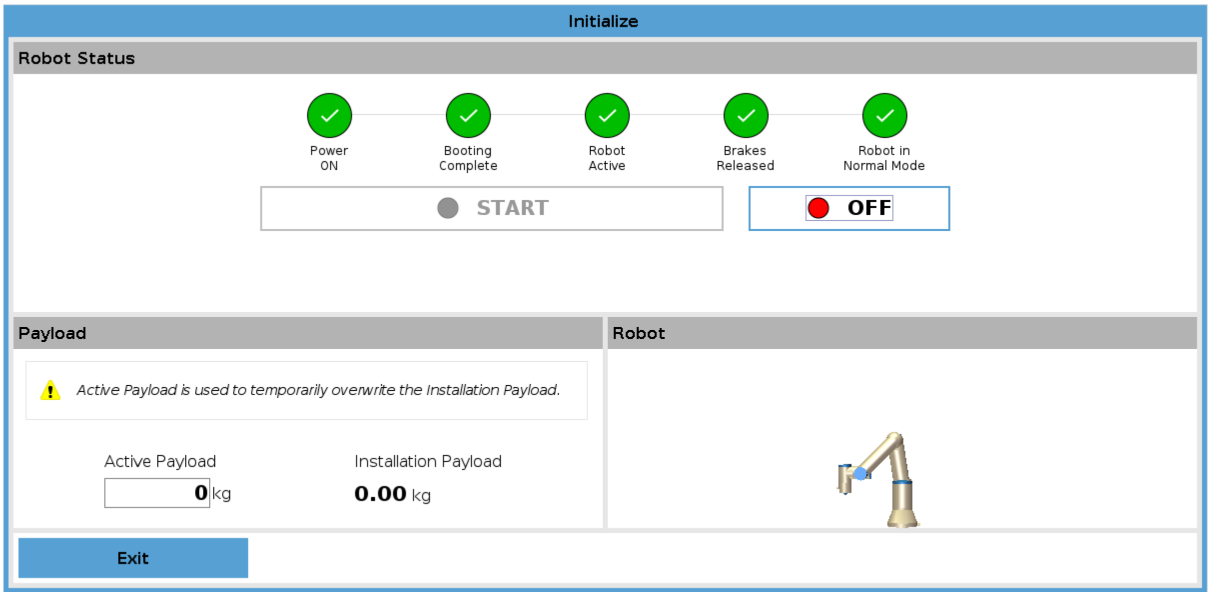
Procedure:

i. **Understand basic operations: Jogging, setting up TCP, CoG, safety features**

Powering up the COBOT

The power up, initializes Polyscope, which will allow us to control the robot. The initialization process switch on the power supply of th**e robot and release the brakes**.

- ❖ Switch on the main power supply, and press the power button on the teach pendant.
- ❖ Go to initialize screen on the teach pendent.
- ❖ In the screen robot status is “POWER OFF” condition. Press the “ON” button in the screen.
- ❖ Now the robot status is “ROBOT IDLE”. Now press the “START” button.
- ❖ After pressing the start button the robot status will be changed. Now the robot status is in “NORMAL MODE”.
- ❖ The robot is ready to do the programming. Robot power on process is completed.
- ❖ Now press the “EXIT” button and enter into the “RUN” mode screen.



Power ON ✓ | Booting Complete ✓ | Robot Active ✓ | Brakes Released ✓ | Robot in Normal Mode ✓

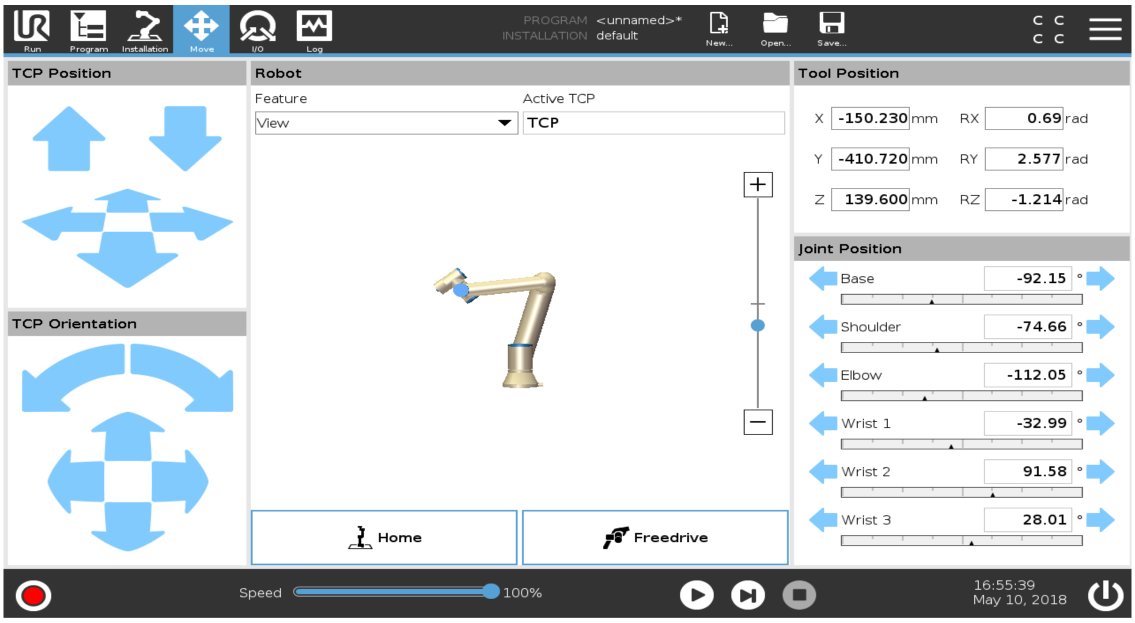
Active Payload is used to temporarily overwrite the Installation Payload.

Active Payload: 0 kg
Installation Payload: 0.00 kg

Fig.8.2: The start screen

Jogging the COBOT

- ❖ Change to Manual mode and enter the password.
- ❖ Goto 'Move' tab and press the arrow keys for jogging.



PROGRAM <unnamed>*
INSTALLATION default

Feature: View
Active TCP: TCP

X	-150.230 mm	RX	0.69 rad		
Y	-410.720 mm	RY	2.577 rad		
Z	139.600 mm	RZ	-1.214 rad		

Base	-92.15 °
Shoulder	-74.66 °
Elbow	-112.05 °
Wrist 1	-32.99 °
Wrist 2	91.58 °
Wrist 3	28.01 °

Speed 100%

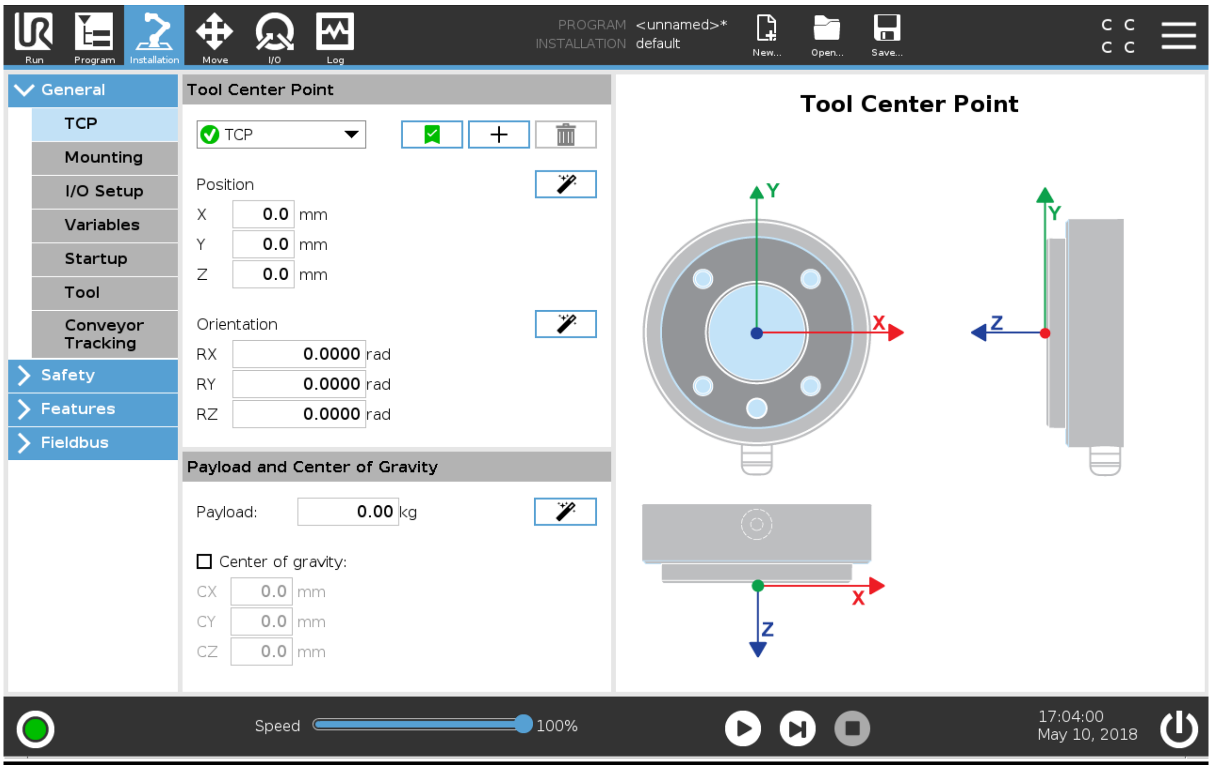
16:55:39
May 10, 2018

Fig.8.3 Move tab to jo the COBOT

<u>**Setting up TCP and CoG**</u>

- ❖ Goto ‘Installation’ tab
- ❖ Under ‘General’, select ‘TCP’
- ❖ Insert values from datasheet or measured values of the following:
  - TCP Position
  - TCP orientation
  - Tool Payload
  - Tool Center of Gravity

In case the values are unknown, the wizard can be used to find the above.



Tool Center Point

TCP

Position
X 0.0 mm
Y 0.0 mm
Z 0.0 mm

Orientation
RX 0.0000 rad
RY 0.0000 rad
RZ 0.0000 rad

Payload and Center of Gravity

Payload: 0.00 kg

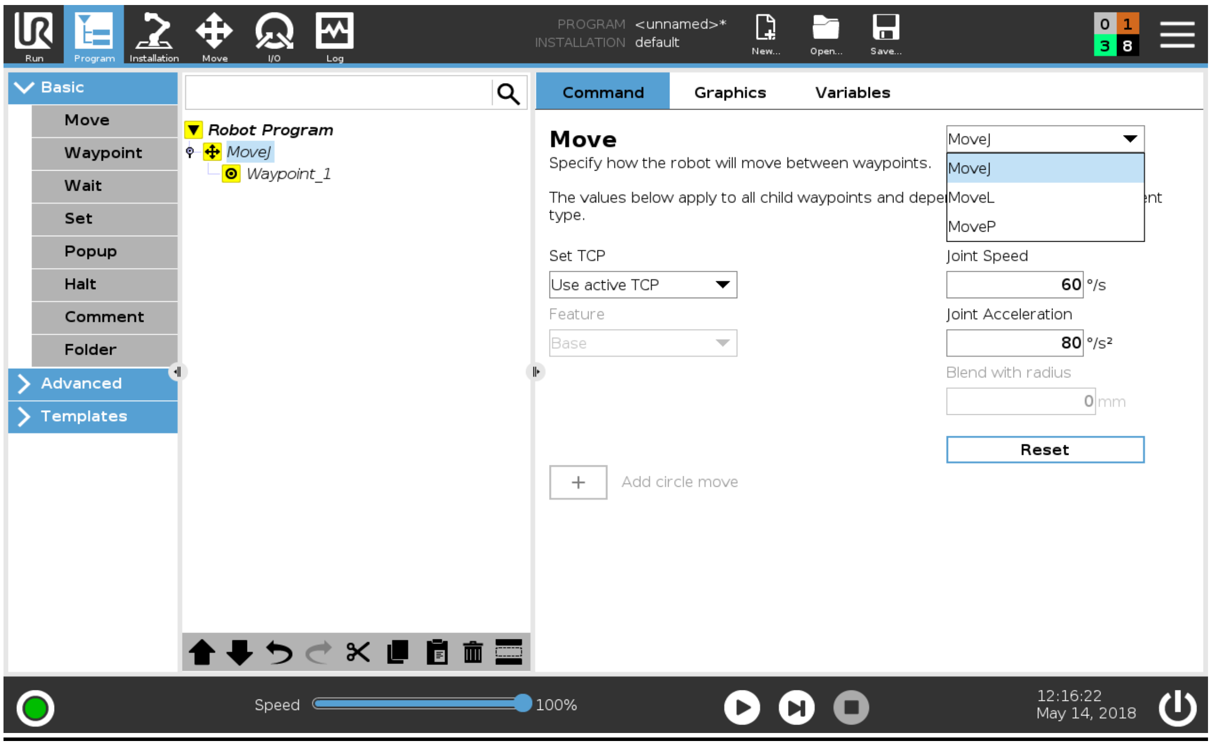
☐ Center of gravity:
CX 0.0 mm
CY 0.0 mm
CZ 0.0 mm

Speed 100%

Fig. 8.4 Installation tab to define TCP

<u>**Writing a new program:**</u>

- ❖ In Run mode top of the screen, there is a “NEW” option. Move the pointer over thereand select the “NEW PROGRAM”.
- ❖ The program tree appears on the screen as per image given below.
- ❖ Select instructions from the options on the left to write the program
- ❖ Now we need to save the program in a specific name. “SAVE” option is next to the ‘new’, and select to “SAVE ALL”.



PROGRAM <unnamed>*
INSTALLATION default

Robot Program
MoveJ
Waypoint_1

Move
Specify how the robot will move between waypoints.

The values below apply to all child waypoints and dependent type.

Set TCP: Use active TCP
Feature: Base
Joint Speed: 60 °/s
Joint Acceleration: 80 °/s²
Blend with radius: 0 mm

Speed 100%

12:16:22
May 14, 2018

Fig. 8.5: Writing a program

Basic commands:

- ❖ Move - Movement types
  - Move J (Joint Movement)
    No interpolation
    Fastest move type
    Useful in "free" space movements
  - Move L (Linear movement)
    Interpolation on
    Linear trajectory for TCP
  - Move P (Process movement)
    Continuous movement
    No stop in Waypoint
    Waypoint 2 (r=25)
- ❖ Waypoint - types
  - Fixed (default)
  - Relative
  - Variable
- ❖ WAIT
  This command should trigger the robot to next operation in the program.

❖ SET
The action you wish the robot to perform at this point in the program. Also user specify changes in the robot payload.

❖ POPUP
User define a message, warning, and user defined input variable can be usingthis command.

❖ HALT
Used to stop the robot program once the program executes.

ii. Implement Pick and Place Operation through programming with teachpendant

- In the program tree, create a move instruction by the basic structure window. A Waypoint will automatically add when inserting Move command in the program tree.
- Select the waypoint to teach and set the waypoint in the command screen.
- Press the “SET WAYPOINT”button, on which one more screen opens. In that screen, move the cobot to its specific location, and press“OK” to set the waypoint.
- Define waypoint for the pick position
- The COBOT should wait for the ‘component present sensor’ signal. Use the WAIT command to wait for the digital input to go high.
- Set the gripper on with the SET instruction when component is detected. Digital output pin 0 is configured for vacuum suction gripper
- insert the “Wait” command to make delay time to proper holding of the component
- Now, define a waypoint for place position.
- Set the gripper off with the SET instruction

9. Implementation of destacking and stacking operation with UR5 COBOT

Aim: Implement destacking and stacking using Loop operation in UR5 COBOT

Theory:

Whatever the nature of business, it is likely to involve the stacking and storage of goods and materials. The basic material handling and storage systems are common to a wide range of stores and warehouses such as pallets and racking systems. De-stacking is largely the reverse process of stacking. The size and shape of a stack depends on the storage space available and on the size, shape, bulk, weight, rigidity or fragility of the articles to be stored. Fig. 9.1 shows the destacking and stacking units in the lab setup. Six Square materials of thickness 6 mm has to be destacked and stacked at the appropriate locations. A loop instruction can be utilized for the purpose.

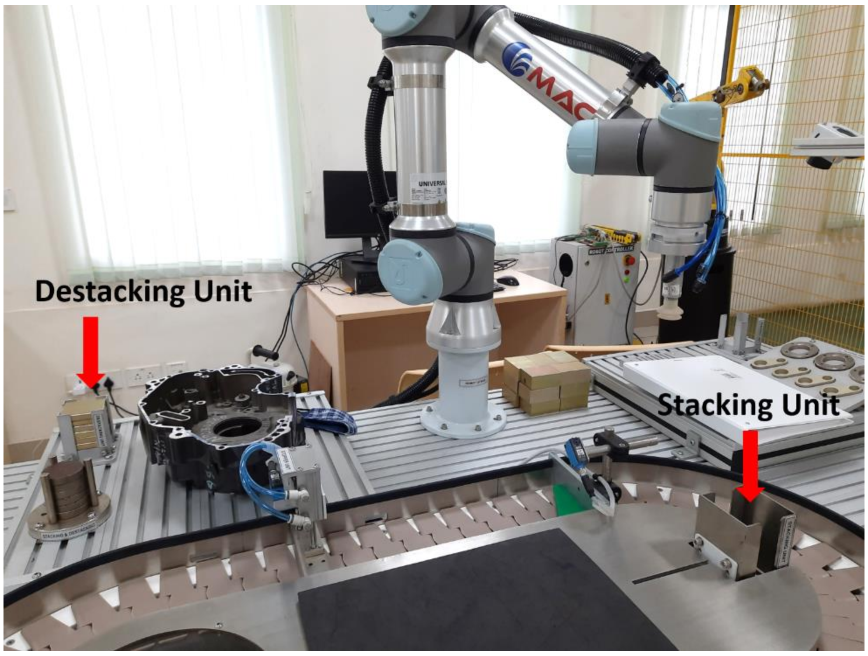


Fig.9.1: The stacking and destacking units

Procedure:

The procedure is for moving the TCP 6mm in the negative Z direction at every iteration:

- Add a before start sequence for assignment of variables and defining the start position.
- Select ‘assignment’ instruction from ‘advanced’ set of instructions. Set the variable (to keep the loop count) to zero.

- Select MOVEJ instruction and and define the start position
- Select the 'assignment' instruction and get the TCP position from the function get_actual_tcp_pose().

  Posstack:=get_actual_tcp_pose()

- In the Robot Main program:
  - Start the 'Loop' from 'advanced' set of instructions and set it to loop for 6 times.
  - Select MOVEJ command. The new position along the negative z axis (current pose + 6mm) has to be defined and stored.
  - Use the assignment operator to define a pose to be added to the start pose:

    Posadd:=p[0,0,-0.006*var_1,0,0,0]

    The six positions are x,y, z, roll, pitch and yaw.

  - Use the assignment operator to add the current pose with Posadd to define the new pose. Use the pose_add function.

    Posnew:=pose_add(Posstack, Posadd)

  - Select Waypoint, set it to variable and select the variable Posnew. This means the robot will now move to the position Posnew, which is offset by 6mm in negative Z from the current position.
  - Increment the variable by 1.
  - Add a Wait command

10. Read DH-parameters and obtain the forward transformation matrix (Using Python)

Aim: Read DH-parameters and obtain the forward transformation matrix (Using Python)

Theory:

DH-Parameters of the 2DoF manipulator manipulator configuration shown is given as-

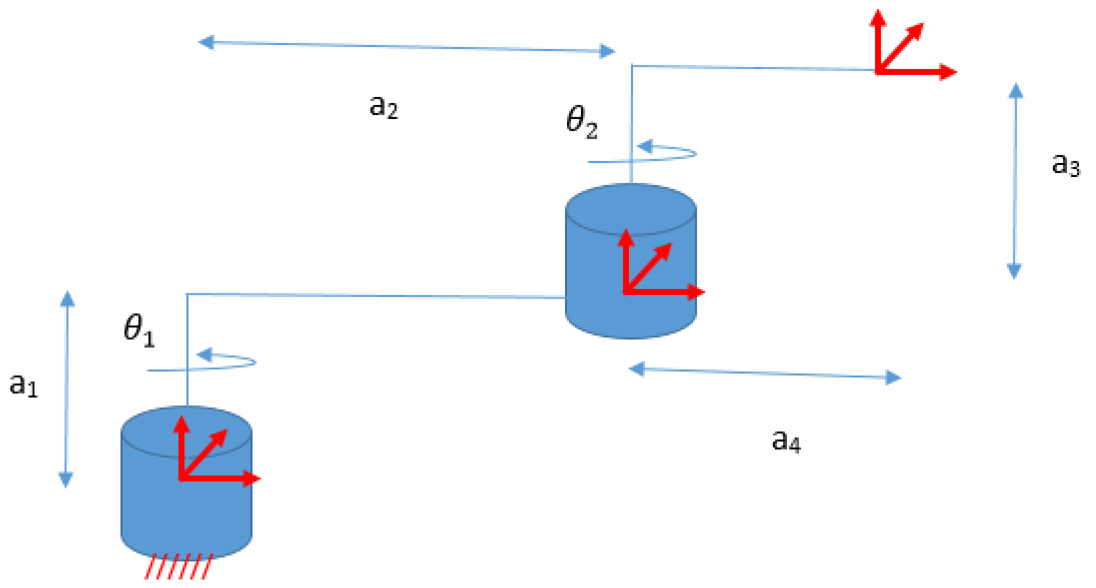


Table 1 DH Parameters

n	ϴ$_n$	α$_n$	r$_n$	d$_n$
1	θ_1	0	a_2	a_1
2	θ_2	0	a_4	a_3

Transformation Matrices 1

$$H_1^0 = \begin{bmatrix} C\theta_1 & -S\theta_1 & 0 & a_2C\theta_1 \\ S\theta_1 & C\theta_1 & 0 & a_2S\theta_1 \\ 0 & 0 & 1 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_2^1 = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & a_4C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & a_4S\theta_2 \\ 0 & 0 & 1 & a_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_2^0 = H_1^0 * H_2^1$$

DH Parameters for Script 1

n	ϴ$_n$	α$_n$	r$_n$	d$_n$
1	90	0	1	1
2	90	0	1	1

1. Students Will be taking different sets of values for $a_1, a_2, a_3, a_4, \theta_1, \theta_2$

	a_1	a_2	a_3	a_3	θ_1	θ_2
SET1	1	3	5	7	30	60
SET2	2	4	6	8	60	90
SET3	1	4	3	5	45	60
SET4	2	5	3	7	30	45

2. Calculate the Transformation matrix H_1^0, H_2^1, H_2^0 manually.
3. Write a python script to calculate H_1^0, H_2^1, H_2^0
4. Verify if manual and software based matrices are same.

Script for reference:

```
import numpy as np
T1=90 #theta 1 angle in degrees
T2=90 #theta 2 angle in degrees
a1=1
a2=1
a3=1
a4=1
d1=a1
d2=a3
T1=(T1/180.0)*np.pi #theta 1 angle in radians
T2=(T2/180.0)*np.pi #theta 2 angle in radians

#Parameter Table
PT=[[(T1/180.0)*np.pi,(0/180.0)*np.pi,a2,a1],
   [(T2/180.0)*np.pi,(0/180.0)*np.pi,a4,a3]]

i=0
H0_1=[[np.cos(PT[i][0]),-np.sin(PT[i][0]),0,PT[i][2]*np.cos(PT[i][0])],
      [np.sin(PT[i][0]),np.cos(PT[i][0]),0,PT[i][2]*np.sin(PT[i][0])],
      [0,0,1,PT[i][3]],
      [0,0,0,1]]
i=1
H1_2=[[np.cos(PT[i][0]),-np.sin(PT[i][0]),0,PT[i][2]*np.cos(PT[i][0])],
      [np.sin(PT[i][0]),np.cos(PT[i][0]),0,PT[i][2]*np.sin(PT[i][0])],
      [0,0,1,PT[i][3]],
      [0,0,0,1]]
```

```
print("H0_1= ")
print (np.matrix(H0_1))
print("H1_2= ")
print (np.matrix(H1_2))

H0_2=np.dot(H0_1,H1_2)
print("H0_2= ")
print (np.matrix(H0_2))
```

Useful Syntax for Raspberry Pi Programing:

Python commands

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD) or GPIO.setmode(GPIO.BCM)

GPIO.setup(7,GPIO.OUT)

GPIO.setup(7,GPIO.IN)

GPIO.output(7,True)

GPIO.output(7,False)

time.sleep(2)

GPIO.output(7,True)

a= GPIO.input(11)

- **PWM wave generation:**

p = GPIO.PWM(12, 50)

p.start(dc) //dc is dutycycle

p.ChangeDutyCycle(dc)

p.stop()

- **Decision Making**

```
if expression:
   statement(s)
else:
   statement(s)
```

- **Loop statement**

```
for iterating_var in sequence:
   statements(s)

eg: for num in range(10,20):

while expression:
   statement(s)
```

ANNEXURE II

Connecting to Raspberry Pi

Using SSH and VNC

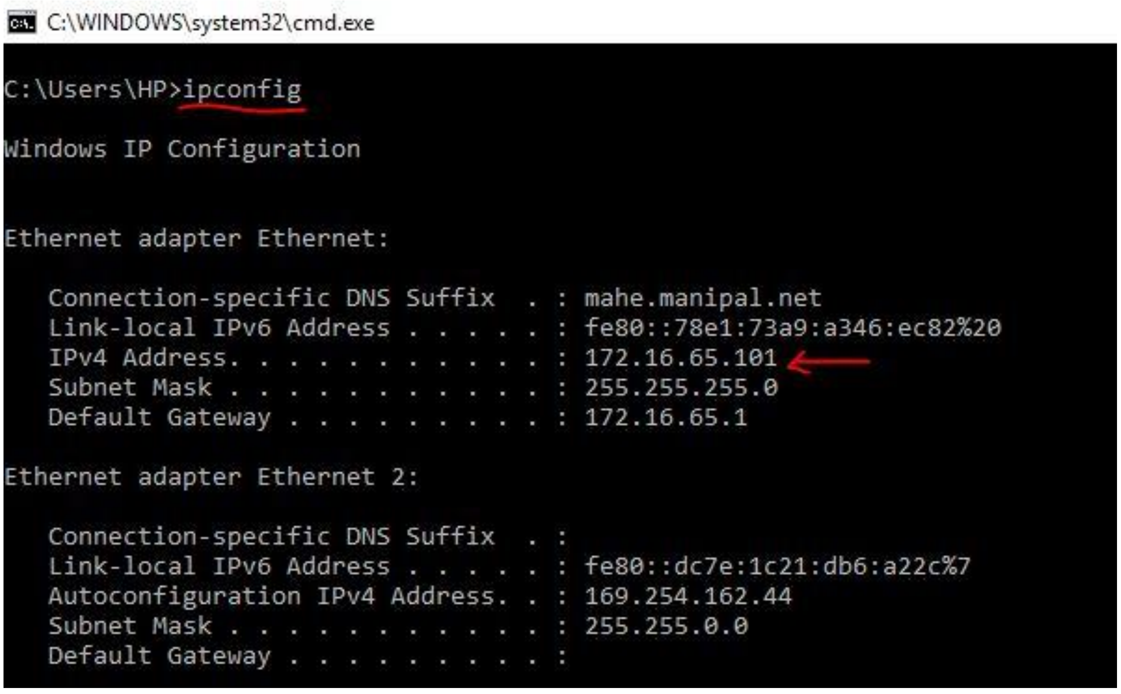
Steps

Step 1: Power RPi and connect to network

Connect your Raspberry Pi to the network and power it. Ensure that the confirmation lights on the Ethernet port of the Raspberry Pi are blinking to indicate a healthy network status

Step 2: Discover your PC's IP address

- **Open the program "cmd" by either using the keyboard shortcut** *windows + R* **and then** *cmd* **or running the** *command prompt* **application**
- **After the window opens, run the command** *ipconfig* **to get the IP address and subnet mask of your system, which must be a system on the same network the Raspberry Pi is connected to**



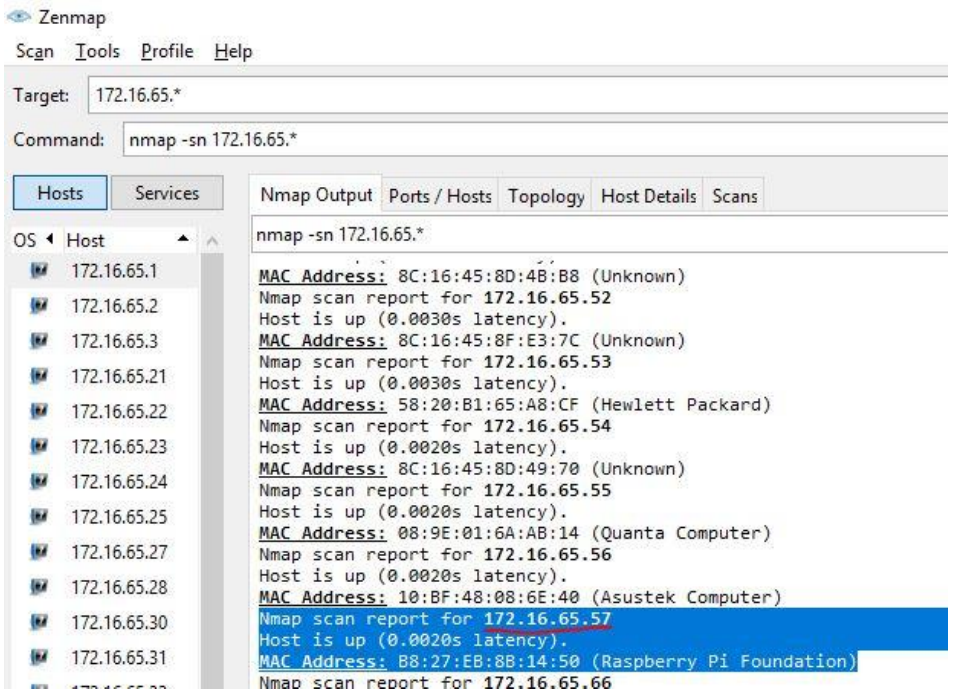
```
C:\WINDOWS\system32\cmd.exe

C:\Users\HP>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : mahe.manipal.net
   Link-local IPv6 Address . . . . . : fe80::78e1:73a9:a346:ec82%20
   IPv4 Address. . . . . . . . . . . : 172.16.65.101
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 172.16.65.1

Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::dc7e:1c21:db6:a22c%7
   Autoconfiguration IPv4 Address. . : 169.254.162.44
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . :
```

- **The IP address of nodes on the network is determined by using the IPv4 address [1] and the Subnet Mask [2] of the network. The IP addresses of nodes on this network will be given by** *172.16.65.** **, this will be our search range.**

Step 3: SSH into RPi using PuTTY

- SSH [3] stands for Secure Shell and is a protocol to access remote terminals
- Run the program *Nmap – Zenmap GUI* (or you could use the *nmap* command) [4]

- In the command, type "*nmap –sn 172.16.65.**", target should be *172.16.65.** and then press scan (or press enter) inspect the output.



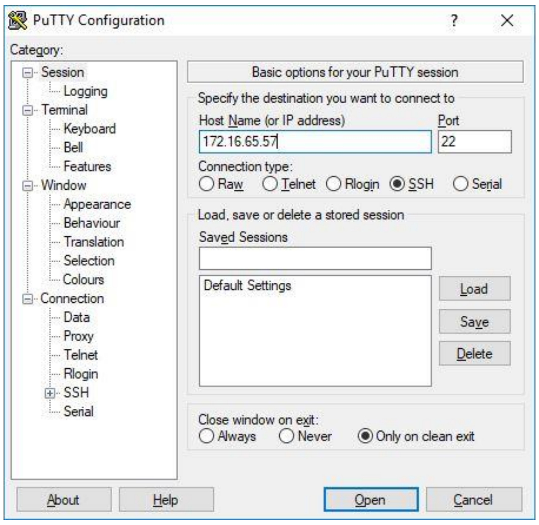
Target: 172.16.65.*

Command: nmap -sn 172.16.65.*

```
172.16.65.1
172.16.65.2
172.16.65.3
172.16.65.21
172.16.65.22
172.16.65.23
172.16.65.24
172.16.65.25
172.16.65.27
172.16.65.28
172.16.65.30
172.16.65.31
```

```
nmap -sn 172.16.65.*

MAC Address: 8C:16:45:8D:4B:B8 (Unknown)
Nmap scan report for 172.16.65.52
Host is up (0.0030s latency).
MAC Address: 8C:16:45:8F:E3:7C (Unknown)
Nmap scan report for 172.16.65.53
Host is up (0.0030s latency).
MAC Address: 58:20:B1:65:A8:CF (Hewlett Packard)
Nmap scan report for 172.16.65.54
Host is up (0.0020s latency).
MAC Address: 8C:16:45:8D:49:70 (Unknown)
Nmap scan report for 172.16.65.55
Host is up (0.0020s latency).
MAC Address: 08:9E:01:6A:AB:14 (Quanta Computer)
Nmap scan report for 172.16.65.56
Host is up (0.0020s latency).
MAC Address: 10:BF:48:08:6E:40 (Asustek Computer)
Nmap scan report for 172.16.65.57
Host is up (0.0020s latency).
MAC Address: B8:27:EB:8B:14:50 (Raspberry Pi Foundation)
Nmap scan report for 172.16.65.66
```

- As seen in the image above, the IP address of the Raspberry Pi is 172.16.65.57. We'll be using this for the entirety of this procedure.
- **Tip:** If there are more than one Raspberry Pi nodes on the network then disconnect the RPi, rescan, note the IP addresses of already existing devices, then connect RPi and rescan and note the IP address of the new RPi added (it'll have a new IP address)
- Open an app called PuTTY [5] type in the IP address. Let the socket number be 22. Connect to the network



Specify the destination you want to connect to

Host Name (or IP address): 172.16.65.57

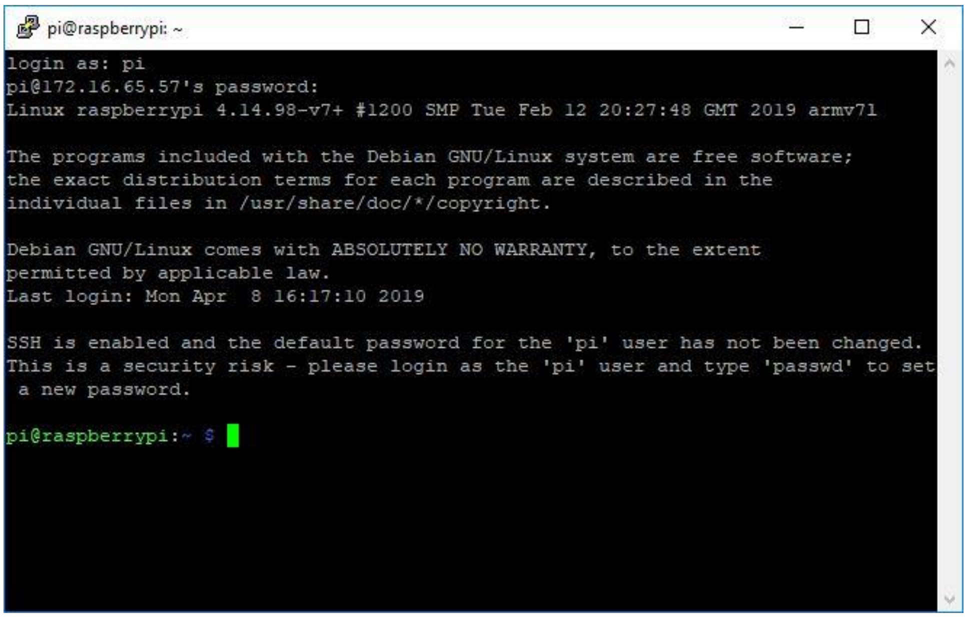
Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Close window on exit: Always Never [✓ Only on clean exit]

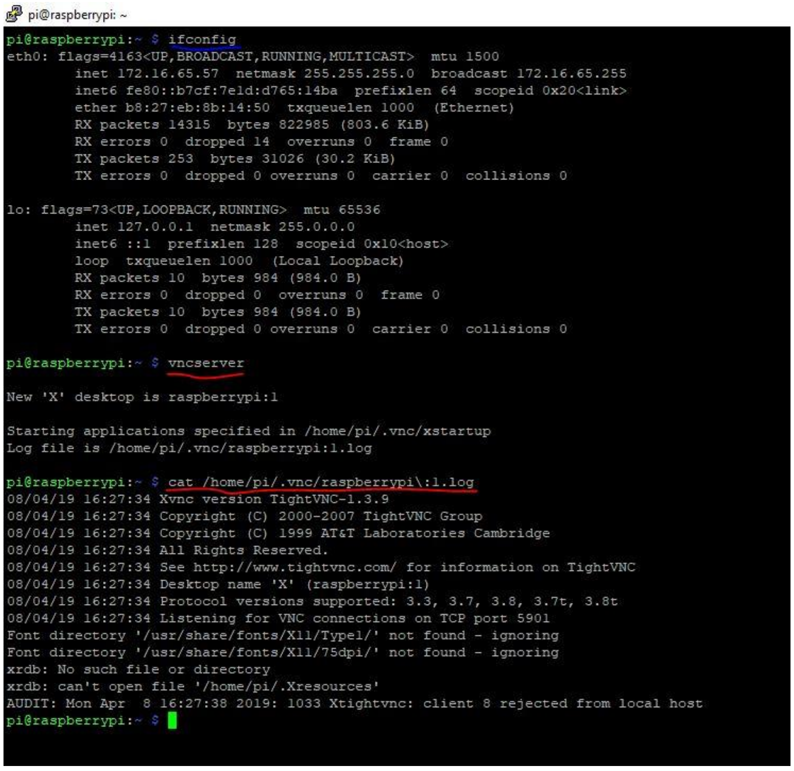
- Click Open and then click Yes if warned about the connection security
- A terminal session will open. This terminal session is running on the RPi.
- Login to the session using the following properties:
  - Id: *pi*
  - Password: *raspberry*

- A terminal session will start with the bash prompt asking you to enter commands



```
login as: pi
pi@172.16.65.57's password:
Linux raspberrypi 4.14.98-v7+ #1200 SMP Tue Feb 12 20:27:48 GMT 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Apr  8 16:17:10 2019

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
 a new password.

pi@raspberrypi:~ $ 
```

- Run the command *ifconfig* to know the IP address of the Raspberry Pi
- Run the command *vncserver* to start the VNC [6] server on the Raspberry Pi
- Optionally, you can check the contents of the log file to know the status of the VNC server session that you just started.

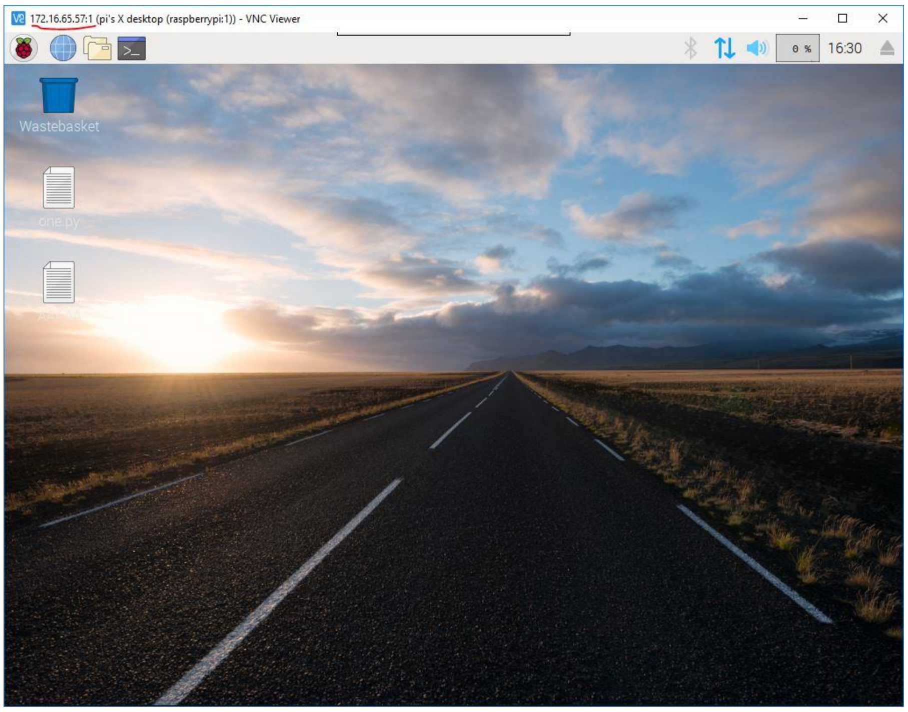


```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.65.57  netmask 255.255.255.0  broadcast 172.16.65.255
        inet6 fe80::b7cf:7e1d:d765:14ba  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:8b:14:50  txqueuelen 1000  (Ethernet)
        RX packets 14315  bytes 822985 (803.6 KiB)
        RX errors 0  dropped 14  overruns 0  frame 0
        TX packets 253  bytes 31026 (30.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 10  bytes 984 (984.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 984 (984.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pi@raspberrypi:~ $ vncserver

New 'X' desktop is raspberrypi:1

Starting applications specified in /home/pi/.vnc/xstartup
Log file is /home/pi/.vnc/raspberrypi:1.log

pi@raspberrypi:~ $ cat /home/pi/.vnc/raspberrypi\:1.log
08/04/19 16:27:34 Xvnc version TightVNC-1.3.9
08/04/19 16:27:34 Copyright (C) 2000-2007 TightVNC Group
08/04/19 16:27:34 Copyright (C) 1999 AT&T Laboratories Cambridge
08/04/19 16:27:34 All Rights Reserved.
08/04/19 16:27:34 See http://www.tightvnc.com/ for information on TightVNC
08/04/19 16:27:34 Desktop name 'X' (raspberrypi:1)
08/04/19 16:27:34 Protocol versions supported: 3.3, 3.7, 3.8, 3.7t, 3.8t
08/04/19 16:27:34 Listening for VNC connections on TCP port 5901
Font directory '/usr/share/fonts/X11/Type1/' not found - ignoring
Font directory '/usr/share/fonts/X11/75dpi/' not found - ignoring
xrdb: No such file or directory
xrdb: can't open file '/home/pi/.Xresources'
AUDIT: Mon Apr  8 16:27:38 2019: 1033 Xtightvnc: client 8 rejected from local host
pi@raspberrypi:~ $ 
```

- The VNC server has been started at *raspberrypi:1*, this means that the VNC Client will have to connect to *172.16.65.57:1* to access the desktop session on the Raspberry Pi.

Step 4: Connect to RPi using VNC

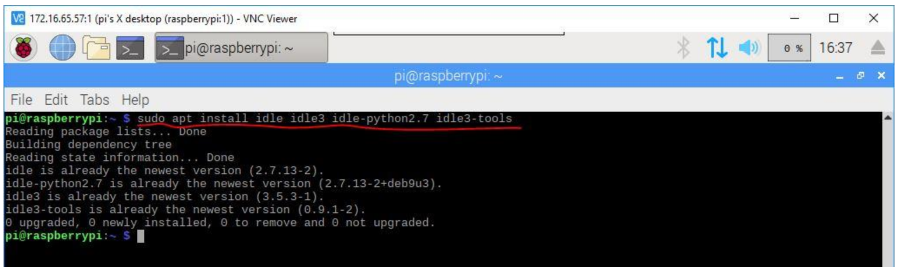
- Open VNC Viewer application and connect to the IP address *172.16.65.57:1*, press Enter after that.
- Click Connect to access the remote desktop


172.16.65.57:1 (pi's X desktop (raspberrypi:1)) - VNC Viewer
16:30
Wastebasket
one.py

- This is now like any other desktop session where you can navigate, create and execute files.

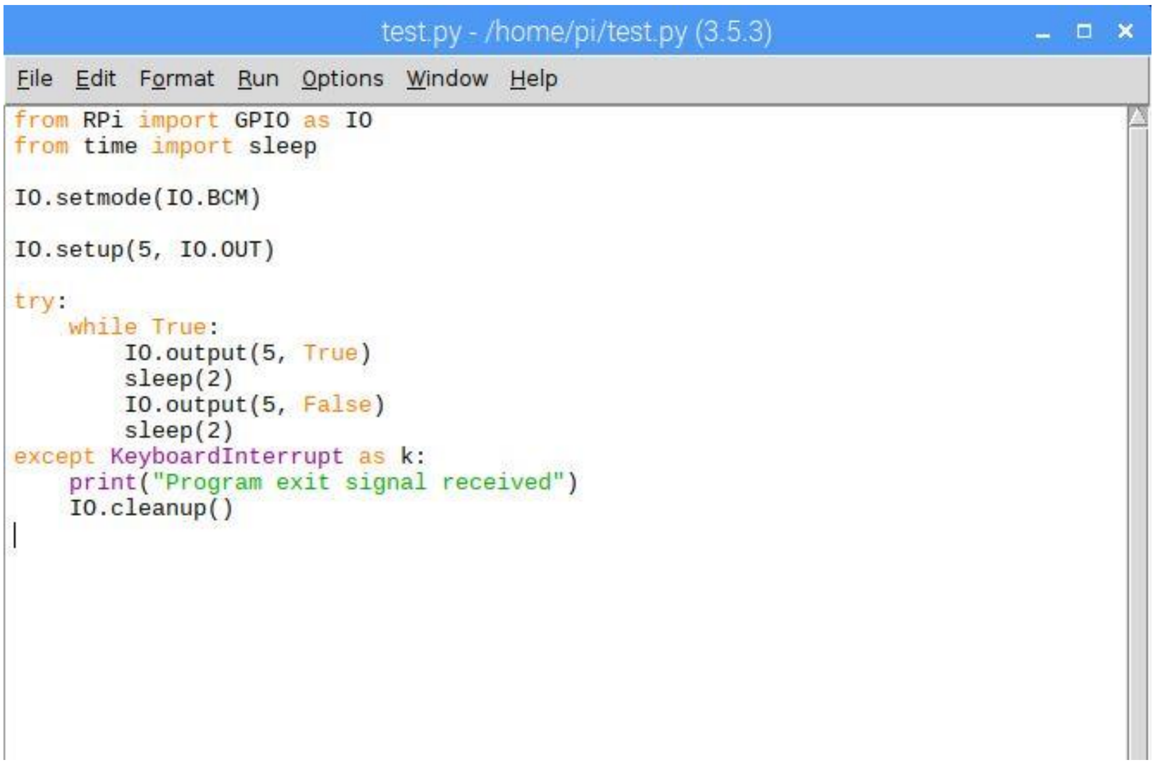
Step 5: Running programs

- Run the command idle for Python 2.7 or idle3 for Python 3.
    - If a window opens, then you're good to go ahead
    - If a window does not open and the command gives you an error telling that the program wasn't found or isn't installed, then run the command *sudo apt install idle idle3 idle-python2.7 idle3-tools* and wait for the command to finish installing the tools.


172.16.65.57:1 (pi's X desktop (raspberrypi:1)) - VNC Viewer
pi@raspberrypi: ~
16:37
File Edit Tabs Help

```
pi@raspberrypi:~ $ sudo apt install idle idle3 idle-python2.7 idle3-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
idle is already the newest version (2.7.13-2).
idle-python2.7 is already the newest version (2.7.13-2+deb9u3).
idle3 is already the newest version (3.5.3-1).
idle3-tools is already the newest version (0.9.1-2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $
```

- After that, you can open a new window and write a Python program. This one will blink an LED on *GPIO5* (pin number 29), with 2 second HIGH and 2 second LOW. The program is shown below.


test.py - /home/pi/test.py (3.5.3)
File Edit Format Run Options Window Help

```python
from RPi import GPIO as IO
from time import sleep

IO.setmode(IO.BCM)

IO.setup(5, IO.OUT)

try:
    while True:
        IO.output(5, True)
        sleep(2)
        IO.output(5, False)
        sleep(2)
except KeyboardInterrupt as k:
    print("Program exit signal received")
    IO.cleanup()
```

- Connect an LED to the GPIO5, run this program and see it blink

References

[1] IP addressing

https://en.wikipedia.org/wiki/IP_address

[2] SSH

https://en.wikipedia.org/wiki/Secure_Shell

[3] Subnet Masks

https://www.iplocation.net/subnet-mask

[4] nmap

https://nmap.org

[5] PuTTY

https://www.putty.org

[6] VNC on Raspberry Pi

https://www.realvnc.com/en/raspberrypi/

https://en.wikipedia.org/wiki/Virtual_Network_Computing

[7] Raspberry Pi

https://www.raspberrypi.org