CPS Lab Programs

Program 1:
Using ESP8266 to calculate the distance of an obstacle using an ultrasonic sensor. If distance calculated is less than certain threshold, it should turn on buzzer. Distance should be displayed on OLED.

1. Select the NodeMCU 1.0 (ESP-12E Module) board from Tools
2. Select the corresponding port which is active
3. By clicking on the second icon in sidebar (Boards Manager), install esp8266
4. By clicking on the third icon in sidebar (Library Manager), install Adafruit GFX and Adafruit SSD1306 libraries

```
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

const int trigPin = D5; //GPIO14
const int echoPin = D6; //GPIO12
const int buzzer = D4; //GPIO2

const int THRESHOLD = 20; //in cm
const float SPEED_OF_SOUND = 0.0343; //cm per microseconds

void setup(){
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(buzzer, OUTPUT);
    Serial.begin(115200);

    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)){
        while(true);
    }
    display.clearDisplay();
}
```

```
void loop(){
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    long duration = pulseIn(echoPin, HIGH);
    float distance = (duration * SPEED_OF_SOUND)/2;

    display.clearDisplay();

    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0,0);
    display.print("Dist:");

    display.setCursor(0,30);
    display.print(distance);
    display.print(" cm");

    if(distance > 0 && distance < THRESHOLD){
        digitalWrite(buzzer, HIGH);
        display.setCursor(0,50);
        display.print("ALERT!");
    }
    else{
        digitalWrite(buzzer, LOW);
    }

    display.display();
    delay(500);
}
```

Connections -

Ultrasonic Pins	ESP8266 Pins
VCC	Vin (5V)
GND	GND
TRIG	D5 (GPIO14)
ECHO	D6 (GPIO12)

OLED Pins	ESP8266 Pins
VCC	3V3 (3.3V)
GND	GND
SCL	D1 (GPIO5)
SDA	D2 (GPIO4)

Buzzer Pins	ESP8266 Pins
VCC/+	D4 (GPIO2)
GND/-	GND

Program 2:
Using ESP8266 calculate the humidity and temperature from the environment and display it on OLED.

1. DHT11 Sensor is used for measuring temperature and humidity
2. Select the NodeMCU 1.0 (ESP-12E Module) board from Tools
3. Select the corresponding port which is active
4. By clicking on the second icon in sidebar (Boards Manager), install esp8266
5. By clicking on the third icon in sidebar (Library Manager), install Adafruit GFX, Adafruit SSD1306 and DHT Sensor library by Adafruit

```
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

#define DHTPIN D7 //GPIO13
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup(){
    Serial.begin(115200);
    dht.begin();

    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)){
```

```
        while(true);
    }
    display.clearDisplay();
    display.setTextColor(SSD1306_WHITE);
}

void loop(){
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();

    if(isnan(humidity)||isnan(temperature)){
        display.clearDisplay();
        display.setTextSize(1);
        display.setCursor(0,0);
        display.print("DHT Error");
        display.display();
        delay(2000);
        return;
    }

    display.clearDisplay();
    display.setTextSize(2);

    display.setCursor(0,0);
    display.print("T: ");
    display.print(temperature);
    display.print(" C");

    display.setCursor(0,30);
    display.print("H: ");
    display.print(humidity);
    display.print(" %");

    display.display();
    delay(2000);
}
```

Connections -

DHT11 Pins	ESP8266 Pins
VCC	3V3 (3.3V)
GND	GND
DATA	D7 (GPIO13)

OLED Pins	ESP8266 Pins
VCC	3V3 (3.3V)
GND	GND
SCL	D1 (GPIO5)
SDA	D2 (GPIO4)

All important commands to remember for Raspberry Pi

1. To check the date

```
date
```

2. To set/change the date and time - If this is not fixed, packages might not be installed as pip gives SSL errors.

```
sudo date -s "YYYY-MM-DD HH:MM:SS"
sudo date -s "2026-01-03 17:45:30"
```

3. Installation and Updating system packages (apt is used for OS-level packages, which is managed by Raspberry Pi OS)

```
sudo apt update
sudo apt upgrade -y
```

4. Installing Python 3 and pip (will already be installed)

```
sudo apt install python3 -y
sudo apt install python3-pip -y
sudo apt install python3-venv -y
```

5. To create and activate a Virtual Environment

```
python3 -m venv env
source env/bin/activate
mkdir folder_name
cd folder_name
nano file_name.py #opens the file
python3 file_name.py #to compile and run the program
```

```
deactivate #to come out of the virtual environment
```

6. GPIO and Visualisation libraries

```
sudo apt install python3-rpi.gpio -y
sudo apt install python3-matplotlib -y
sudo apt install python3-pandas -y
```

- To install Python only libraries use pip3/pip
- pip may install for Python 2 as well (if it exists on the system), hence use pip3 as its safer and installs only for Python 3
- To install system wide libraries which have to interact with hardware as well use sudo apt (ex: GPIO)

Program 3:
Using a Raspberry Pi, collect data using temperature sensor and apply data visualisation techniques.

1. Create a virtual environment, and type the code

```
python3 -m venv env
source env/bin/activate
nano file_name.py
```

```
import time
import board
import adafruit_dht
import matplotlib.pyplot as plt

dht = adafruit_dht.DHT11(board.D4)

timestamps = []
temperatures = []
humidities = []

for i in range(10):
    try:
        current_time = time.strftime("%H:%M:%S")
        temp = dht.temperature
        humi = dht.humidity

        timestamps.append(current_time)
```

```
        temperatures.append(temp)
        humidities.append(humi)

        time.sleep(2)

    except RuntimeError as error:
        print(error)
        time.sleep(2)

plt.plot(timestamps, temperatures, label="Temp (C)", marker='o')
plt.plot(timestamps, humidities, label="Humidity (%)", marker='x')
plt.xlabel("Time")
plt.ylabel("Values")
plt.title("Temperature and Humidity variation")
plt.legend()
plt.xticks(rotation = 45)
plt.tight_layout()
plt.show()
```

2. Running the program

```
python3 file_name.py

#if the above command does not work because it could not access GPIO, try this
sudo python3 file_name.py

#good practice to always check this in case of errors
sudo apt update

#necessary packages needed for this program are:
sudo apt install python3-matplotlib -y
pip3 install adafruit-blinka #board module
pip3 install adafruit-circuitpython-dht #adafruit_dht module
```

Connections:

Power cable (C-type)

Remove monitor's HDMI, and connect the HDMI to micro-HDMI cable of Raspberry Pi

Connect the keyboard and mouse to USB ports

Remove ethernet cable from CPU and connect to Raspberry Pi

DHT11 Pins	Raspberry Pi Pins
VCC	5V (Pin no. 2/4)
GND	GND (Pin no. 6)

DHT11 Pins	Raspberry Pi Pins
DATA	D4/GPIO4 (Pin no. 7)

Program 4:
Using a Raspberry Pi, collect data using a temperature sensor and upload it to a cloud platform (ThingSpeak).

To set up a ThingSpeak account:

- Open the website and login with your MATLAB account
- Create a new channel
- Enable Field 1 (rename to temperature) and 2 (rename to humidity), and save the channel
- Go to the API Keys section and copy the WRITE API KEY
- The graphs will be shown in Private View once the program is executed

1. Create a virtual environment and type the code

```
python3 -m venv env
source env/bin/activate
nano file_name.py
```

```
import time
import board
import adafruit_dht
import requests

dht = adafruit_dht.DHT11(board.D4)

API_KEY = "YOUR_WRITE_API_KEY"
THINGSPEAK_URL = "https://api.thingspeak.com/update"

while True:
    try:
        temperature = dht.temperature
        humitdity = dht.humidity
        payload = {
            'api_key':API_KEY,
            'field1':temperature,
            'field2':humidity
        }
        response = requests.get(THINGSPEAK_URL, params = payload)
        print("Uploaded Temp:",temperature,"Humidity:",humidity)
```

```
        time.sleep(15)

    except RuntimeError as e:
        print("Sensor error: ",e)
        time.sleep(2)

    except Exception as e:
        print("Upload error: ",e)
        time.sleep(5)
```

2. Necessary packages needed for this program are

```
sudo apt update

pip3 install requests #HTTP Library required for cloud communication
pip3 install adafruit-blinka #board module
pip3 install adafruit-circuitpython-dht #for DHT sensor

#to run the program
python3 file_name.py
```

Connections:
Power cable (C-type)
Remove monitor's HDMI, and connect the HDMI to micro-HDMI cable of Raspberry Pi
Connect the keyboard and mouse to USB ports
Remove ethernet cable from CPU and connect to Raspberry Pi

DHT11 Pins	Raspberry Pi Pins
VCC	5V (Pin no. 2/4)
GND	GND (Pin no. 6)
DATA	D4/GPIO4 (Pin no. 7)

Program 5:
Using Raspberry Pi to control servo motor.

1. Create a virtual environment and type the code

```
python3 -m venv env
source env/bin/activate
nano file_name.py
```

```
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

SERVO_PIN = 18 #GPIO18
GPIO.setup(SERVO_PIN, GPIO.OUT)
pwm = GPIO.PWM(SERVO_PIN, 50)
pwm.start(0)

def set_angle(angle):
    min_duty = 2.5
    max_duty = 12.5
    duty = min_duty + (max_duty - min_duty) * (angle / 180)

    pwm.ChangeDutyCycle(duty)
    time.sleep(1)
    pwn.ChangeDutyCycle(0)

try:
    while True:
        user_input = input("Enter angle between 0-180 or q to exit: ")

        if(user_input.lower() == 'q'):
            print("Exiting...")
            break
        try:
            angle = float(user_input)
            if(angle<0 or angle>180):
                print("Invalid input)
            else:
                print(f"Rotating servo by {angle} degrees")
                set_angle(angle)
        except ValueError:
            print("Invalid input")

finally:
    pwm.stop()
    GPIO.cleanup()
```

2. Necessary packages needed

```
sudo apt update
```

```
sudo apt install python3-rpi.gpio -y #for importing RPi.GPIO module

python3 file_name.py #to run the program
```

Connections:

Power cable (C-type)

Remove monitor's HDMI, and connect the HDMI to micro-HDMI cable of Raspberry Pi

Connect the keyboard and mouse to USB ports

Remove ethernet cable from CPU and connect to Raspberry Pi

Servo Motor	**Raspberry Pi Pins**
RED (VCC)	5V (Pin no. 2/4)
BROWN (GND)	GND (Pin no. 6)
ORANGE (SIGNAL)	GPIO18 (Pin no. 12)

Program 8:

Using Raspberry Pi, show the communication between a client and server.

1. Create a virtual environment and create two files

```
#server.py

import socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('0.0.0.0',12345))

sever_socket.listen(1)
print("Server listening on port 12345...")

client_socket, client_address = server_socket.accept()
print("Connected to client: ",client_address)

message = client_socket.recv(1024)
print("Message from client: ",message.decode())
client_socket.send("Message received by server".encode())

client_socket.close()
server_socket.close()
print("Server connection closed")
```

```
#client.py

import socket
SERVER_IP = '127.0.0.1'
SERVER_PORT = 12345

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(SERVER_IP, SERVER_PORT)
print("Connected to server")

client_socket.send("Hi from client".encode())
message = client_socket.recv(1024)
print("Message received: ",message.decode())

client_socket.close()
print("Client connection closed")
```

2. Open two terminals. Run the server.py in the first terminal and client.py in the second terminal simultaneously. IMPORTANT - Run the server.py first or there will be an error.

```
#terminal 1
python3 server.py

#terminal 2
python3 client.py
```

All important commands to remember for Jetson Nano

1. Creating a virtual environment. IMP: It is not necessary to create a virtual environment, and the code might not run. Deactivate the virtual environment and then run the program again.

```
python3 -m venv env
source env/bin/activate
nano file_name.py
```

2. Installing packages needed for Jetson nano programs

```
sudo apt update
sudo apt upgrade -y

sudo apt install python3-rpi.gpio -y
```

```
#FOR cv2 module
sudo apt install python3-opencv -y

#FOR notify2 module which is used for notification services
sudo apt install python3-notify2 -y
sudo apt install libnotify-bin -y
```

Program 6:
Using Jetson Nano, capture a live image with the help of USB camera and send it as notification.

```
import cv2
import notify2
import os
import sys

notify2.init("Jetson Nano Camera Notifications")

cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: Could not open camera")
    n = notify2.Notification(
        "Camera error",
        "Could not open USB Camera"
    )
    n.show()
    sys.exit()

ret, frame = cap.read()
if not ret:
    print("Error: Could not read frame")
    n = notify2.Notification(
        "Capture Error",
        "Could not read frame"
    )
    n.show()
    sys.exit()

output_dir = "captured_images"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

image_path = os.path.join(output_dir, "newimage.jpg")
cv2.imwrite(image_path, frame)
```

```
print(f"Image saved at {image_path}")
cap.release()
n = notify2.Notification(
    "Image captured",
    "Image saved successfully"
)
n.show()
```

To run the program:

```
python3 file_name.py
```

If there are any errors or any modules not found while running the program, install the packages mentioned above outside the virtual environment.

Connections:
Power cable - is connected to CPU USB port
HDMI cable is removed from the CPU and connected to Jetson Nano (unlike RPi where it is removed from monitor)
Keyboard, Mouse and Camera are connected to USB ports of Jetson Nano

Program 7:
Using Jetson Nano, capture a live image with the help of USB camera and send it as notification. Also, annotate the image.

```
import cv2
import notify2
import os
import sys

notify2.init("Jetson Nano Camera Notification")

cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Could not open USB Camera")
    n = notify2.Notification(
        "Camera error",
        "Could not open camera"
    )
    n.show()
    sys.exit()

ret, frame = cap.read()
```

```
if not ret:
    print("Could not capture frame")
    n = notify2.Notification(
        "Capture error",
        "Could not capture frame"
    )
    n.show()
    sys.exit()

x = 100
y = 100
w = 200
h = 150
cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0), 2)

output_dir = "captured_images"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

image_path = os.path.join(output_dir, "newimage.jpg")
cv2.imwrite(image_path, frame)
cap.release()

print(f"Image saved at {image_path}")
n = notify2.Notification(
    "Image captured",
    "Image saved with ROI successfully"
)
n.show()
```

Connections:

Power cable - is connected to CPU USB port
HDMI cable is removed from the CPU and connected to Jetson Nano (unlike RPi where it is removed from monitor)
Keyboard, Mouse and Camera are connected to USB ports of Jetson Nano

THEORY FOR VIVA

- Define CPS, Sensors, Actuators, Embedded System

```
CPS is defined as the system that links the cyber world and physical world,
through a network of interconnected elements like sensors, actuators.
Sensors are those devices which sense the physical environment and collect the
data.
Actuators are those which act on the environment, by converting electrical
```

```
energy to other forms (mech/sound/light).
Embedded system refers to a combination of microprocessor, I/O devices and
memory.
```

- Arduino UNO uses ATmega328P microcontroller. It has 14 digital and 6 analog I/O pins.
- Node MCU (Micro Controller Unit) is an open-source development built on SoC (System on Chip) called ESP8266.
- Raspberry Pi has its own OS.
- Jetson Nano is an embedded SoM (System on Module), which is running on Ubuntu Linux.

Program - 1 and 2

1. Why are each of the modules imported?
   Arduino.h - core Arduino functions
   Wire.h - enable I2C communication, to communicate with devices using SCA & SDL lines
   Adafruit_GFX.h - core graphics functions
   Adafruit_SSD1306.h - used to control OLED displays
2. What is the function of trig pin and echo pin?
   Trig Pin is used to send trigger pulses to initiate ultrasonic waves, while echo pin is used to receive the reflected echo signal after it hits the obstacle.
3. Why is the distance divided by 2 in the formula?
   Because it contains the time taken for the wave to travel to the object and back to the sensor. Hence, it is double the time. To get the one way distance, we divide it by 2.
4. What are the functions of OLED and why are they used?
   display.clearDisplay() - it is used to clear the buffer
   display.display() - it is used to update the OLED with new values
   display.setTextColor(), display.setTextSize(), display.setCurosor(), display.print()
5. What does the pulseIn() function return?
   Time in microseconds
6. Why is ESP8266 preferred over Arduino UNO?
   It has in-built Wi-fi module. It is faster and has more memory power.
7. What is I2C communication, and why are the two wires used?
   I2C refers to Inter-Integrated Circuit. It uses two wires SCL (for clock signal) and SDA (for data transfer).
8. What type of sensor is DHT11?
   It is a digital sensor which measures temperature and humidity.

9. What does isnan() function check?
   It checks whether sensor reading is invalid or not a number.
10. Difference between DHT11 and DHT22?
    DHT11 has lower measurement accuracy and has a smaller temperature and humidity range.

Program - 3 and 4

- Raspberry Pi has higher processing power as compared to ESP8266 and can handle data visualisation libraries like matplotlib.
- Why do we create a virtual environment?
  We create virtual environment to install packages only inside venv and prevent version conflicts.
- Why are each of the modules imported?
  board is used to initialize GPIO pins, adafruit_dht is used for the sensor, matplotlib for data visualisation/graphs
- What is the use of time.sleep(2)?
  To delay the sensor readings or it will be too immediate
- What is ThingSpeak and why is used?
  It is a cloud platform, which is used for remote monitoring of data.
  We use a WRITE_API_KEY to upload data onto the platform.
- ThingSpeak allows data updates only once every 15 seconds for free accounts.
  Hence, the delay is time.sleep(15).