Program 1: ESP8266 Ultrasonic + OLED + Buzzer

```
#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;
const int echoPin = D6;
const int buzzer = D4;
const int THRESHOLD = 20;
const float SPEED_OF_SOUND = 0.0343;

void setup(){
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buzzer, OUTPUT);
  Serial.begin(115200);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  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.setCursor(0,0);
  display.print("Dist:");
  display.setCursor(0,30);
  display.print(distance);

  if(distance > 0 && distance < THRESHOLD){
    digitalWrite(buzzer, HIGH);
  } else {
    digitalWrite(buzzer, LOW);
  }

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

Connections:
Ultrasonic:
VCC → Vin
GND → GND
TRIG → D5
ECHO → D6

OLED:
VCC → 3V3
GND → GND
SCL → D1
SDA → D2

Buzzer:
+ → D4
- → GND

Commands:
Select NodeMCU 1.0
Install esp8266 board
Install Adafruit GFX
Install SSD1306

Program 2: ESP8266 DHT11 Temperature & Humidity with OLED

```
#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

#define DHTPIN D7
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

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:
VCC → 3V3
GND → GND
DATA → D7 (GPIO13)

OLED:
VCC → 3V3
GND → GND
SCL → D1 (GPIO5)
SDA → D2 (GPIO4)

Commands:
Select NodeMCU 1.0
Install esp8266 board
Install Adafruit GFX
Install Adafruit SSD1306
Install DHT Sensor Library

Program 3: Raspberry Pi – DHT11 Data Collection & Visualization

```
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()
```

Connections:
DHT11:
VCC → 5V (Pin 2/4)
GND → GND (Pin 6)
DATA → GPIO4 (Pin 7)

Commands:
sudo apt update
sudo apt install python3-matplotlib -y
pip3 install adafruit-blinka
pip3 install adafruit-circuitpython-dht

Run:
python3 file_name.py
(if GPIO error → sudo python3 file_name.py)

Program 4: Raspberry Pi – DHT11 Data Upload to ThingSpeak

```
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
        humidity = 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)
```

Connections:
DHT11:
VCC → 5V (Pin 2/4)
GND → GND (Pin 6)
DATA → GPIO4 (Pin 7)

ThingSpeak:
Create Channel
Enable Field1 (Temp)
Enable Field2 (Humidity)
Copy WRITE API KEY

Commands:
sudo apt update
pip3 install requests
pip3 install adafruit-blinka
pip3 install adafruit-circuitpython-dht

Run:
python3 file_name.py

Program 5: Raspberry Pi – Servo Motor Control using PWM

```
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)  # 50Hz
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)
    pwm.ChangeDutyCycle(0)

try:
    while True:
        user_input = input("Enter angle (0-180) or q to quit: ")
        if user_input.lower() == 'q':
            break
        angle = float(user_input)
        if angle < 0 or angle > 180:
            print("Invalid angle")
        else:
            set_angle(angle)

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

Connections:
Servo Motor:
RED (VCC) → 5V (Pin 2/4)
BROWN (GND) → GND (Pin 6)
ORANGE (SIGNAL) → GPIO18 (Pin 12)

Commands:
sudo apt update
sudo apt install python3-rpi.gpio -y

Run:
python3 file_name.py

Program 6: Jetson Nano – USB Camera Capture with 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)

cap.release()

n = notify2.Notification("Image Captured", "Image saved successfully")
n.show()

print(f"Image saved at {image_path}")
```

Connections:
USB Camera → Jetson USB Port
Keyboard, Mouse → USB
HDMI → Jetson (from CPU)

Commands:
sudo apt update
sudo apt install python3-opencv -y
sudo apt install python3-notify2 -y
sudo apt install libnotify-bin -y

Run:
python3 file_name.py

Program 7: Jetson Nano – USB Camera Capture with ROI Annotation

```
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()

# ROI coordinates
x, y, w, h = 100, 100, 200, 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:
USB Camera → Jetson USB Port
Keyboard, Mouse → USB
HDMI → Jetson (from CPU)

Commands:
sudo apt update
sudo apt install python3-opencv -y
sudo apt install python3-notify2 -y
sudo apt install libnotify-bin -y

Run:
python3 file_name.py

Program 8: Client–Server Communication using Sockets

```
# ---------- server.py ----------
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('0.0.0.0', 12345))
server_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")
```

Concept:
TCP Client–Server Model

Commands:
Terminal 1:
python3 server.py

Terminal 2:
python3 client.py

IMPORTANT:
Run server FIRST

Notes:
AF_INET → IPv4
SOCK_STREAM → TCP
recv(1024) → buffer size