Student name: Zakarya Guerinat

Student ID: 217090531

SIT225: Data Capture Technologies

Activity 2.1: Working with sensor - DHT22

DHT22 is a temperature and humidity sensor.

Hardware Required

Arduino Board

DHT22 sensor

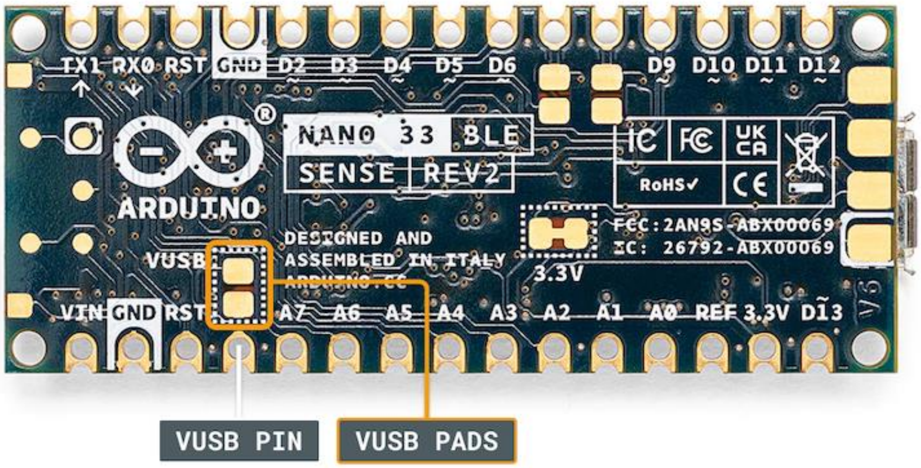
USB cable

Software Required

Arduino programming environment

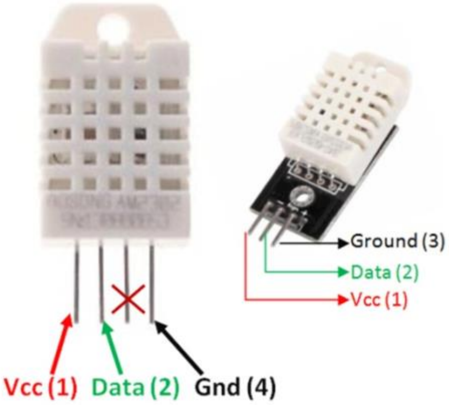
Known issue, action required

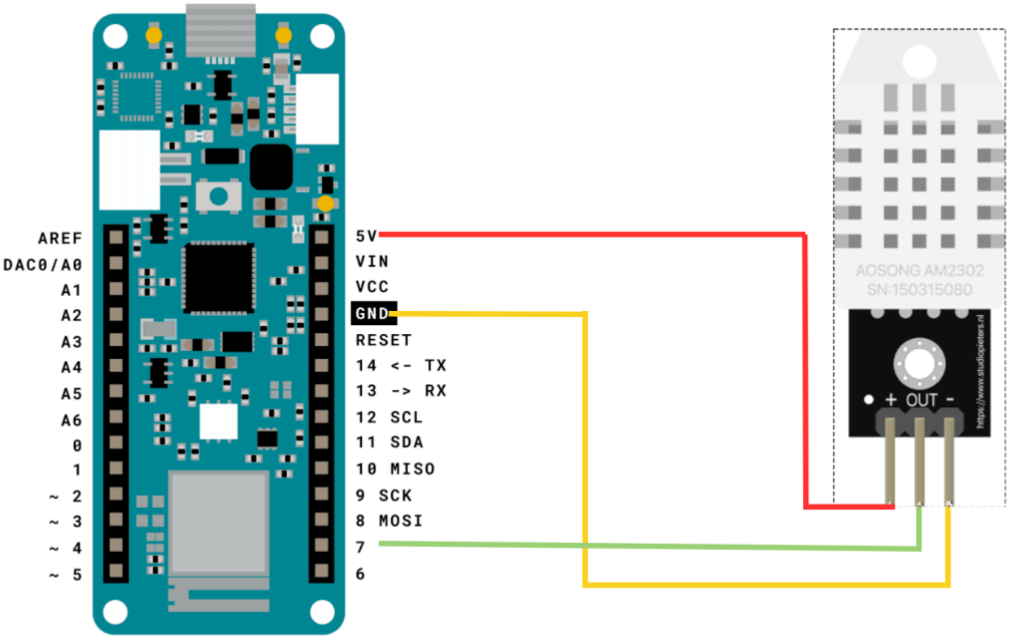
Arduino Nano 33 IoT board operates on 3.3 V, it needs to be arranged to make it 5 V. The Arduino board has a pin called VUSB or VBUS and there are 2 pads next to the pin. **These two pads must be shorted to enable the pin** (see detail here https://support.arduino.cc/hc/en-us/articles/360014779679-Enable-5-V-power-on-the-VUSB-or-VBUS-pin-on-Nano-boards).



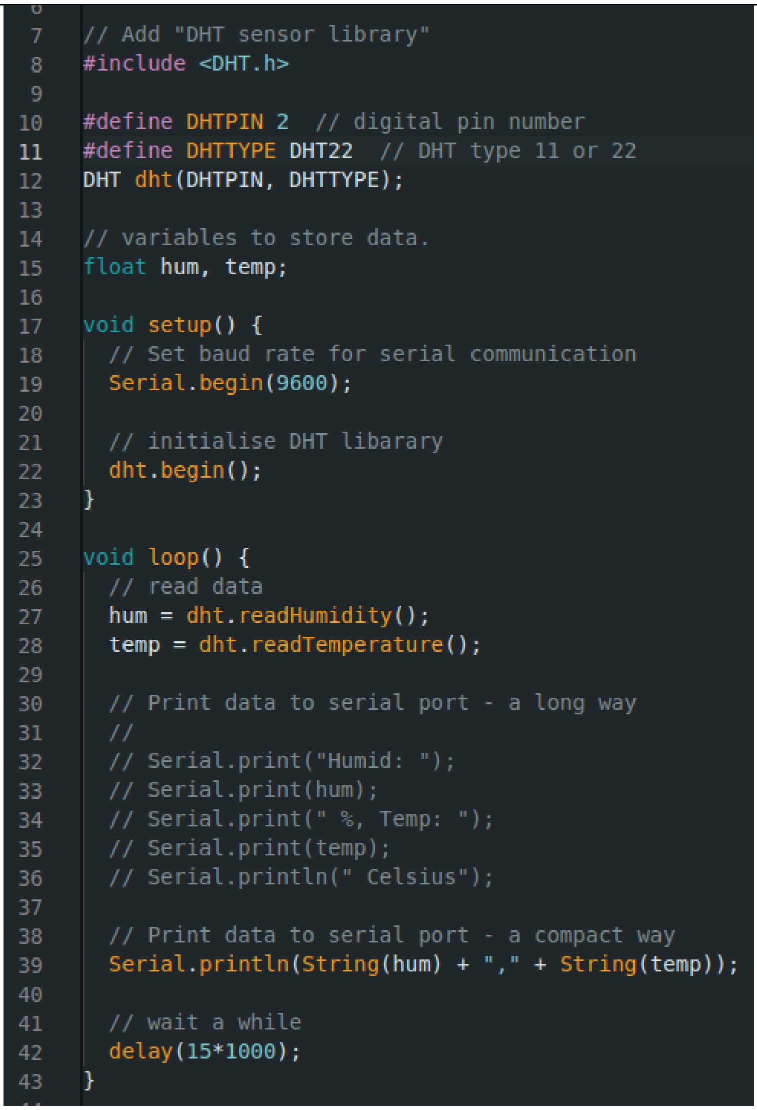
To test, you can use a wire and connect these 2 pads manually by hand to see if data is coming through DHT22 sensor. For long data collection, you will need to solder the wire permanently to connect the pads. Seeking help from tutors there is an on-campus facility called Maker Space where you can do it.

Steps:

Step	Action
1	Connect your DHT22 Temperature and Humidity Sensor to the Arduino board. Note that the pin layout in the image below may look different than the board you may have.  Ground (3) Data (2) Vcc (1) Vcc (1) Data (2) Gnd (4)

	 AREF DAC0/A0 A1 A2 A3 A4 A5 A6 0 1 ~ 2 ~ 3 ~ 4 ~ 5 5V VIN VCC GND RESET 14 <- TX 13 -> RX 12 SCL 11 SDA 10 MISO 9 SCK 8 MOSI 7 6 AOSONG AM2302 SN:150315080 + OUT - https://www.studiopieters.nl a. Pick a red male-female jumper wire and attach the female end to pin 1 (VCC pin) on the sensor. Plug the male end into the Arduino board’s 5V power pin. b. Pick a blue male-female jumper wire and attach the female end to pin 2 (DATA pin) on the sensor. Plug the male end into the Arduino board’s digital data pin 2. c. Pick a black male-female jumper wire and attach the female end to pin 4 (GND) on the sensor. Plug the male end into the Arduino board’s GND pin. d. Sensor’s pin 3 is not used.
2	Connect your Arduino board to your computer using the USB cable.
3	Write an Arduino sketch (or download it from https://github.com/deakin-deep-dreamer/sit225/blob/main/week_2/sketch_dht22.ino) which looks like below. Compile the code in Arduino IDE, deploy to the board and observe output in the Arduino IDE serial monitor.

<table>
<tr>
<td></td>
<td>



```
// Add "DHT sensor library"
#include <DHT.h>

#define DHTPIN 2  // digital pin number
#define DHTTYPE DHT22  // DHT type 11 or 22
DHT dht(DHTPIN, DHTTYPE);

// variables to store data.
float hum, temp;

void setup() {
  // Set baud rate for serial communication
  Serial.begin(9600);

  // initialise DHT libarary
  dht.begin();
}

void loop() {
  // read data
  hum = dht.readHumidity();
  temp = dht.readTemperature();

  // Print data to serial port - a long way
  //
  // Serial.print("Humid: ");
  // Serial.print(hum);
  // Serial.print(" %, Temp: ");
  // Serial.print(temp);
  // Serial.println(" Celsius");

  // Print data to serial port - a compact way
  Serial.println(String(hum) + "," + String(temp));

  // wait a while
  delay(15*1000);
}
```

</td>
</tr>
<tr>
<td>4</td>
<td>

Question: A spec of the DHT22 sensor is given in the link below. It mentions that the sampling rate is 0.5 Hz.

https://lastminuteengineers.com/dht11-dht22-arduino-tutorial

i) What does the sampling rate mean?

ii) Where is this used in the Arduino code?

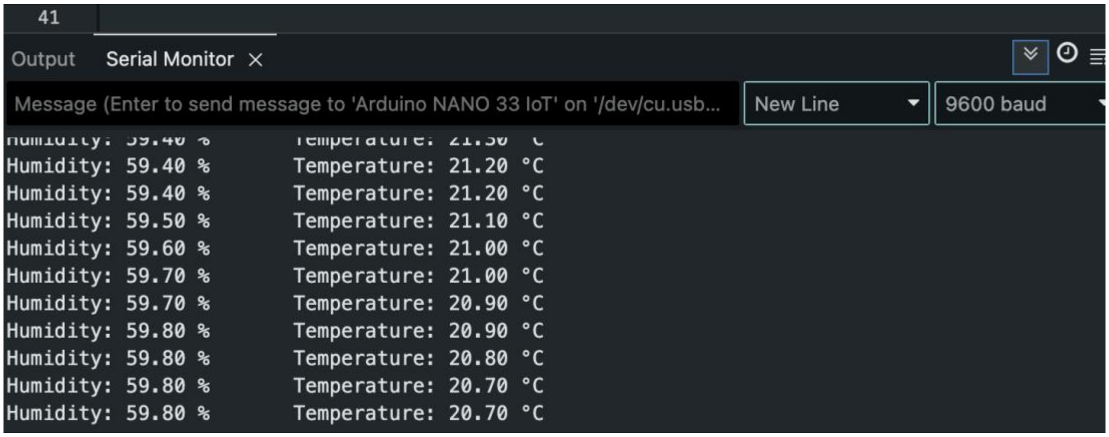
Answer: The sampling rate means how often the Arduino reads data from the sensor,its shown in the code by delay(2000); 2000ms = 2 secs

</td>
</tr>
<tr>
<td>5</td>
<td>

Question: Take a screenshot of your Serial Monitor displaying temperature & humidity sensor data logs. Add the image here.

</td>
</tr>
</table>

Answer:



Message (Enter to send message to 'Arduino NANO 33 IoT' on '/dev/cu.usb...

New Line

9600 baud

```
Humidity: 59.40 %       Temperature: 21.30 °C
Humidity: 59.40 %       Temperature: 21.20 °C
Humidity: 59.40 %       Temperature: 21.20 °C
Humidity: 59.50 %       Temperature: 21.10 °C
Humidity: 59.60 %       Temperature: 21.00 °C
Humidity: 59.70 %       Temperature: 21.00 °C
Humidity: 59.70 %       Temperature: 20.90 °C
Humidity: 59.80 %       Temperature: 20.90 °C
Humidity: 59.80 %       Temperature: 20.80 °C
Humidity: 59.80 %       Temperature: 20.70 °C
Humidity: 59.80 %       Temperature: 20.70 °C
```

Activity 2.2: Working with sensor - HC-SR04

HC-SR04 is an Ultrasonic sensor.

Hardware Required

Arduino Board

HC-SR04 Ultrasonic sensor

USB cable

Software Required

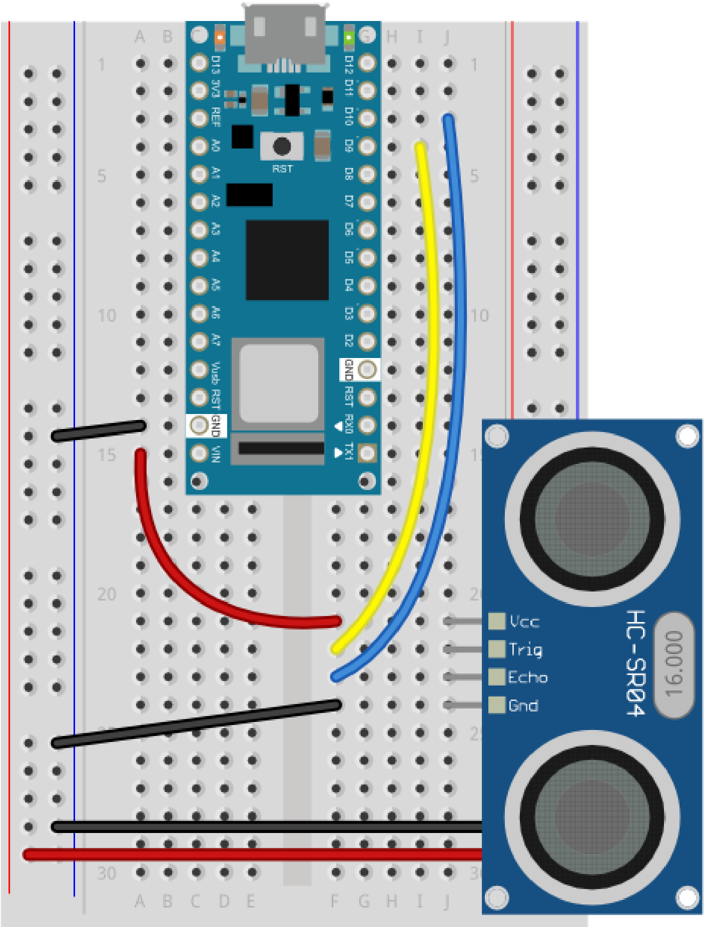
Arduino programming environment

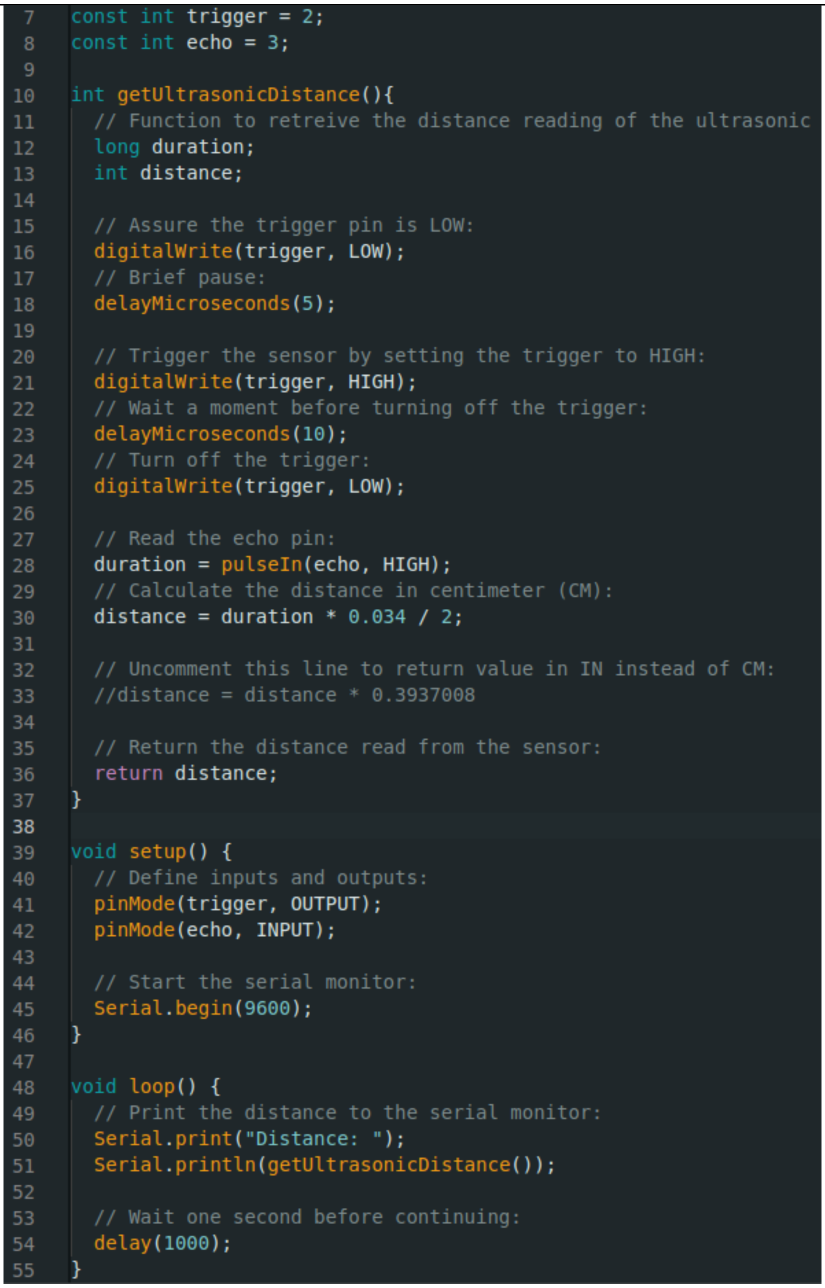
Known issue, action required

The same known issue applies to SR04 which operates at 5 V source while the Arduino Nano 33 IoT supplies 3.3 V. It requires 2 VUSB (or VBUS) pads to be shorted to enable the pin.

Steps:

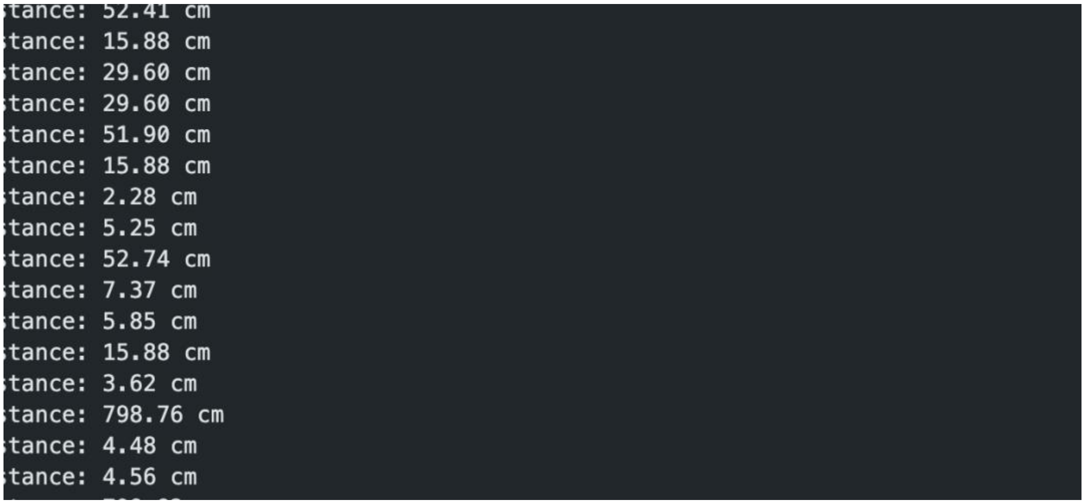
Step	Action
1	Connect your HC-SR04 sensor to the Arduino board. Note that the pin layout in the image below may look different than the board you may have.

	 HC-SR04 16.000 Vcc Trig Echo Gnd RST D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 Vusb RST GND VIN D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1 A B C D E F G H I J 1 5 10 15 20 25 30
2	Connect your Arduino board to your computer using the USB cable.
3	Write an Arduino sketch (or download it from https://github.com/deakin-deep-dreamer/sit225/blob/main/week_2/sketch_hcsr04_distance.ino) which looks like below. Compile the code in Arduino IDE, deploy to the board and observe output in the Arduino IDE serial monitor.



```
const int trigger = 2;
const int echo = 3;

int getUltrasonicDistance(){
  // Function to retreive the distance reading of the ultrasonic
  long duration;
  int distance;

  // Assure the trigger pin is LOW:
  digitalWrite(trigger, LOW);
  // Brief pause:
  delayMicroseconds(5);

  // Trigger the sensor by setting the trigger to HIGH:
  digitalWrite(trigger, HIGH);
  // Wait a moment before turning off the trigger:
  delayMicroseconds(10);
  // Turn off the trigger:
  digitalWrite(trigger, LOW);

  // Read the echo pin:
  duration = pulseIn(echo, HIGH);
  // Calculate the distance in centimeter (CM):
  distance = duration * 0.034 / 2;

  // Uncomment this line to return value in IN instead of CM:
  //distance = distance * 0.3937008

  // Return the distance read from the sensor:
  return distance;
}

void setup() {
  // Define inputs and outputs:
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);

  // Start the serial monitor:
  Serial.begin(9600);
}

void loop() {
  // Print the distance to the serial monitor:
  Serial.print("Distance: ");
  Serial.println(getUltrasonicDistance());

  // Wait one second before continuing:
  delay(1000);
}
```

4	Question: Spec of SR04 is available here (https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf). Identify 2 critical aspects you should be careful about this sensor operation. Answer: 1 always connect the GND wire first before powering it on 2 make sure the object is at least 0.5 square metres in size on a surface that is flat and smooth

5	Question: Take a screenshot of your Serial Monitor displaying distance values while you try to generate a periodic motion by moving your hand gradually back and forth towards the sensor. Add the image here. Answer:  tance: 52.41 cm tance: 15.88 cm tance: 29.60 cm tance: 29.60 cm tance: 51.90 cm tance: 15.88 cm tance: 2.28 cm tance: 5.25 cm tance: 52.74 cm tance: 7.37 cm tance: 5.85 cm tance: 15.88 cm tance: 3.62 cm tance: 798.76 cm tance: 4.48 cm tance: 4.56 cm

Activity 2.3: Working with sensor - Accelerometer

LSM6DS3 module on the Arduino Nano 33 IoT is an accelerometer and gyroscope sensor.

Hardware Required

Arduino Nano 33 IoT Board (has inbuilt LSM6DS3 module)

USB cable

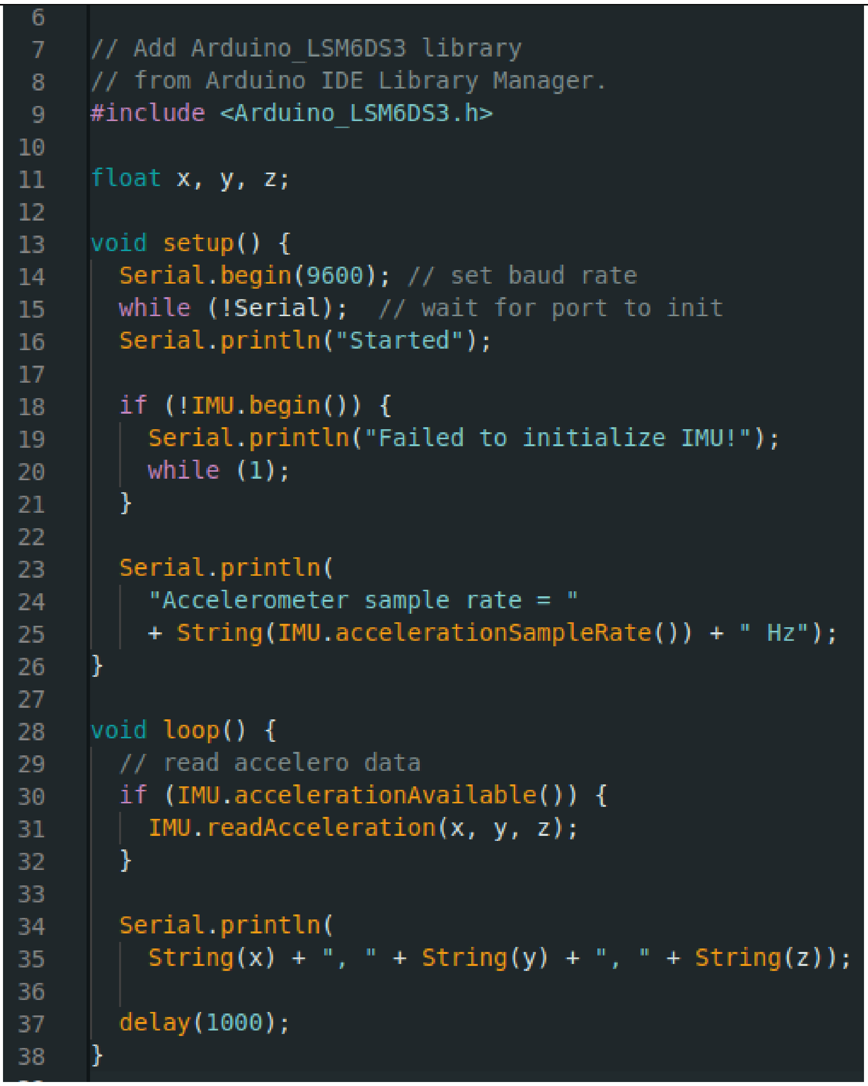
Software Required

Arduino programming environment

Steps:

Step	Action
1	Write Arduino sketch (or download code from https://github.com/deakin-deep-dreamer/sit225/blob/main/week_2/sketch_accelero.ino) which looks like below. Compile the code in Arduino IDE, deploy to the board and observe output in the Arduino IDE serial monitor.

<table>
<tr>
<td></td>
<td>



```
// Add Arduino_LSM6DS3 library
// from Arduino IDE Library Manager.
#include <Arduino_LSM6DS3.h>

float x, y, z;

void setup() {
  Serial.begin(9600); // set baud rate
  while (!Serial);  // wait for port to init
  Serial.println("Started");

  if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");
    while (1);
  }

  Serial.println(
    "Accelerometer sample rate = "
    + String(IMU.accelerationSampleRate()) + " Hz");
}

void loop() {
  // read accelero data
  if (IMU.accelerationAvailable()) {
    IMU.readAcceleration(x, y, z);
  }

  Serial.println(
    String(x) + ", " + String(y) + ", " + String(z));

  delay(1000);
}
```

</td>
</tr>
<tr>
<td>2</td>
<td>

Question: Spec of LSM6DS3 is available here (https://content.arduino.cc/assets/st_imu_lsm6ds3_datasheet.pdf). Identify at least 3 attributes of this sensor you think important to work with.

Answer:

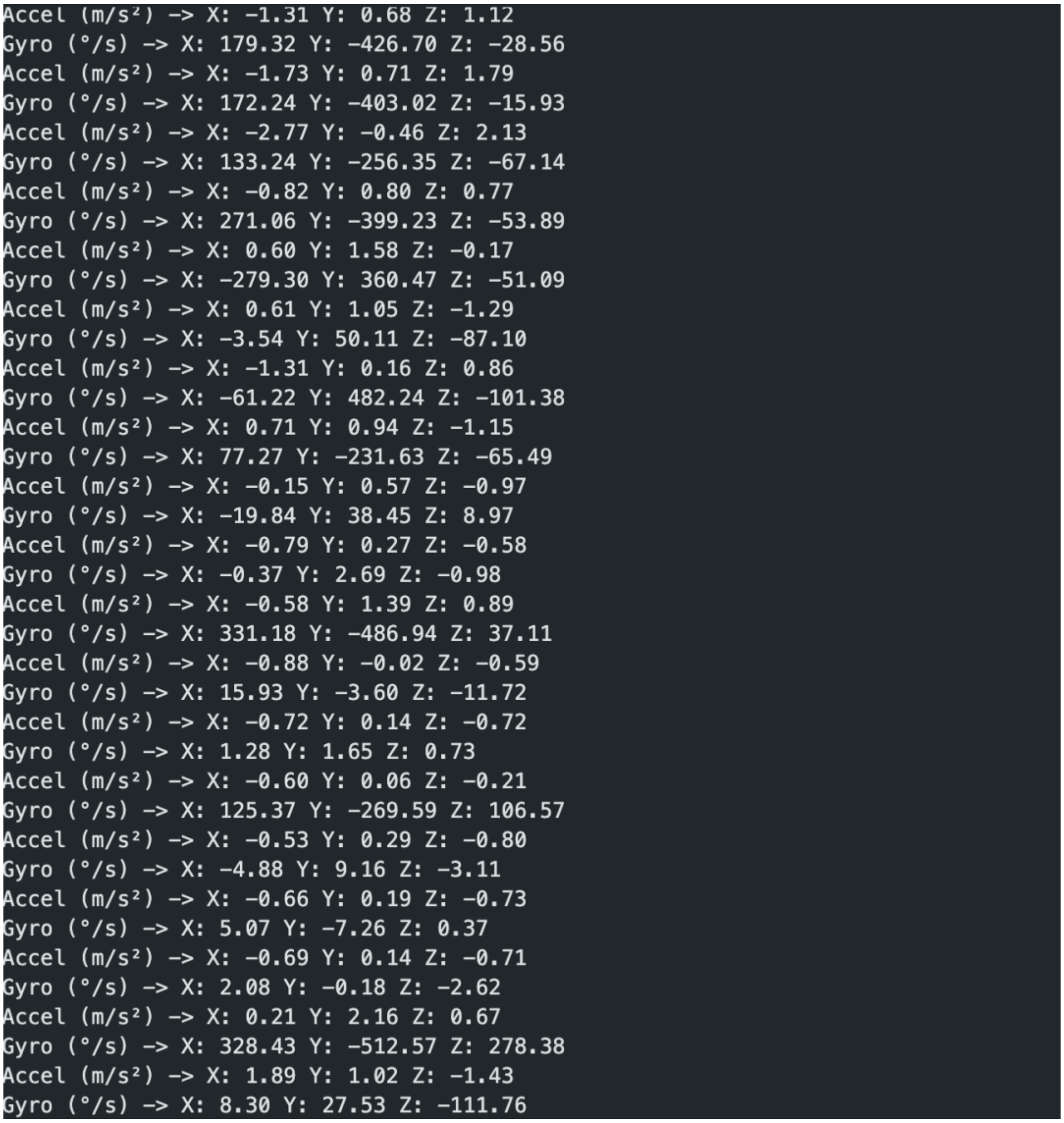
a) Measurment range
b) Sampling rate
c) Interrupt capabilities

</td>
</tr>
<tr>
<td>3</td>
<td>

Question: Take a screenshot of your Serial Monitor displaying sensor readings. Add the image here.

</td>
</tr>
</table>

Answer:



```
Accel (m/s²) -> X: -1.31 Y: 0.68 Z: 1.12
Gyro (°/s) -> X: 179.32 Y: -426.70 Z: -28.56
Accel (m/s²) -> X: -1.73 Y: 0.71 Z: 1.79
Gyro (°/s) -> X: 172.24 Y: -403.02 Z: -15.93
Accel (m/s²) -> X: -2.77 Y: -0.46 Z: 2.13
Gyro (°/s) -> X: 133.24 Y: -256.35 Z: -67.14
Accel (m/s²) -> X: -0.82 Y: 0.80 Z: 0.77
Gyro (°/s) -> X: 271.06 Y: -399.23 Z: -53.89
Accel (m/s²) -> X: 0.60 Y: 1.58 Z: -0.17
Gyro (°/s) -> X: -279.30 Y: 360.47 Z: -51.09
Accel (m/s²) -> X: 0.61 Y: 1.05 Z: -1.29
Gyro (°/s) -> X: -3.54 Y: 50.11 Z: -87.10
Accel (m/s²) -> X: -1.31 Y: 0.16 Z: 0.86
Gyro (°/s) -> X: -61.22 Y: 482.24 Z: -101.38
Accel (m/s²) -> X: 0.71 Y: 0.94 Z: -1.15
Gyro (°/s) -> X: 77.27 Y: -231.63 Z: -65.49
Accel (m/s²) -> X: -0.15 Y: 0.57 Z: -0.97
Gyro (°/s) -> X: -19.84 Y: 38.45 Z: 8.97
Accel (m/s²) -> X: -0.79 Y: 0.27 Z: -0.58
Gyro (°/s) -> X: -0.37 Y: 2.69 Z: -0.98
Accel (m/s²) -> X: -0.58 Y: 1.39 Z: 0.89
Gyro (°/s) -> X: 331.18 Y: -486.94 Z: 37.11
Accel (m/s²) -> X: -0.88 Y: -0.02 Z: -0.59
Gyro (°/s) -> X: 15.93 Y: -3.60 Z: -11.72
Accel (m/s²) -> X: -0.72 Y: 0.14 Z: -0.72
Gyro (°/s) -> X: 1.28 Y: 1.65 Z: 0.73
Accel (m/s²) -> X: -0.60 Y: 0.06 Z: -0.21
Gyro (°/s) -> X: 125.37 Y: -269.59 Z: 106.57
Accel (m/s²) -> X: -0.53 Y: 0.29 Z: -0.80
Gyro (°/s) -> X: -4.88 Y: 9.16 Z: -3.11
Accel (m/s²) -> X: -0.66 Y: 0.19 Z: -0.73
Gyro (°/s) -> X: 5.07 Y: -7.26 Z: 0.37
Accel (m/s²) -> X: -0.69 Y: 0.14 Z: -0.71
Gyro (°/s) -> X: 2.08 Y: -0.18 Z: -2.62
Accel (m/s²) -> X: 0.21 Y: 2.16 Z: 0.67
Gyro (°/s) -> X: 328.43 Y: -512.57 Z: 278.38
Accel (m/s²) -> X: 1.89 Y: 1.02 Z: -1.43
Gyro (°/s) -> X: 8.30 Y: 27.53 Z: -111.76
```

4

Question: Identify the max sampling rate and consider reducing the delay (line 37 in the sketch) to increase the number of samples. Summarise your findings here.

Answer: reducing the delay will increase the frequency of data sampling which will improve responsiveness and resolution.

Activity 2.4: Plot data using Python Notebook

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. You can find detail in official website (https://matplotlib.org).

Hardware Required

No hardware required

Software Required

Python Jupyter notebook

Steps:

Step	Action
1	Download the Jupyter notebook **week2_notebook.ipynb** from here (https://github.com/deakin-deep-dreamer/sit225/blob/main/week_2/week2_notebook.ipynb). Follow the instructions in the notebook to carry out instructions and finally convert the notebook to PDF so you can combine it with this activity sheet PDF.

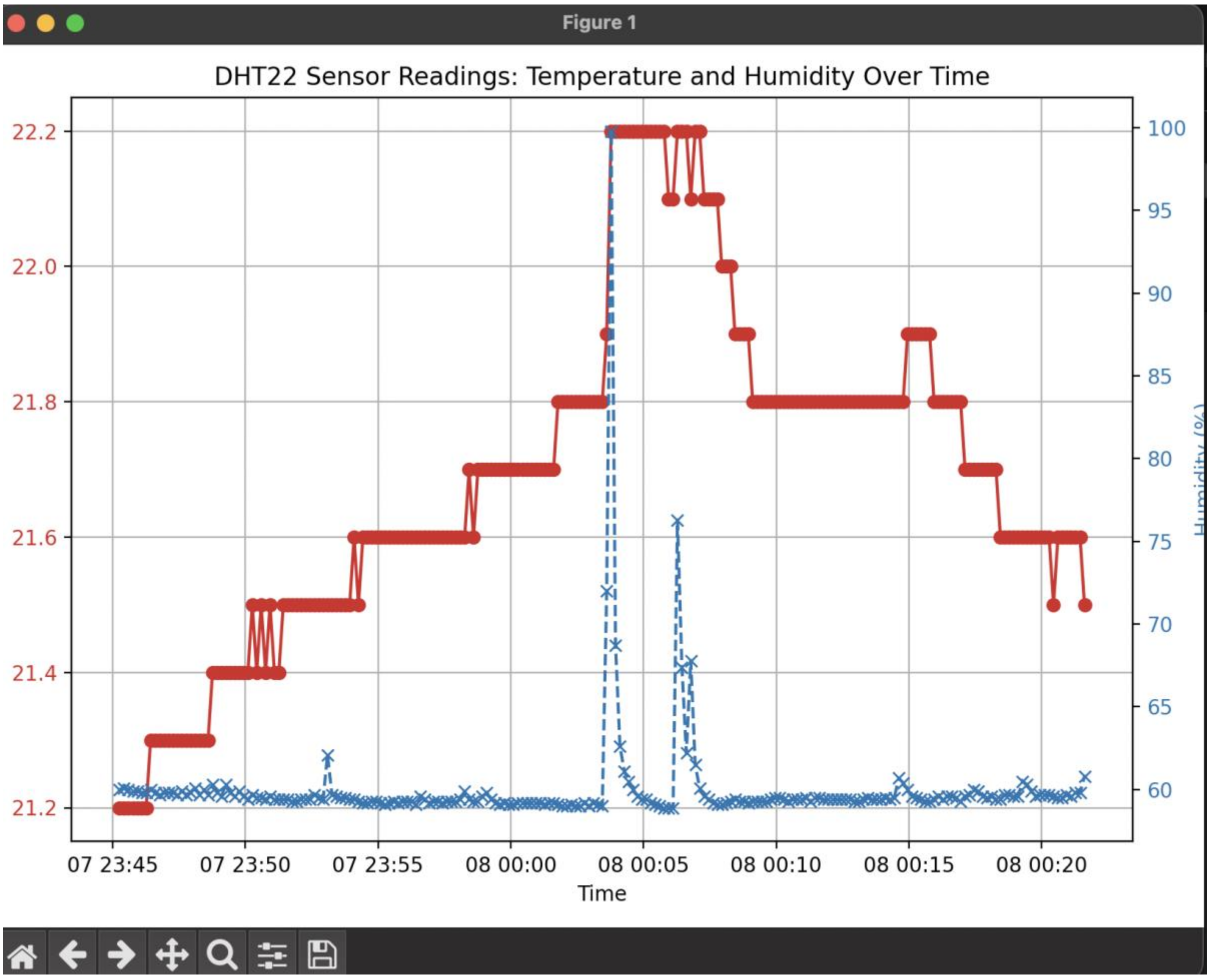
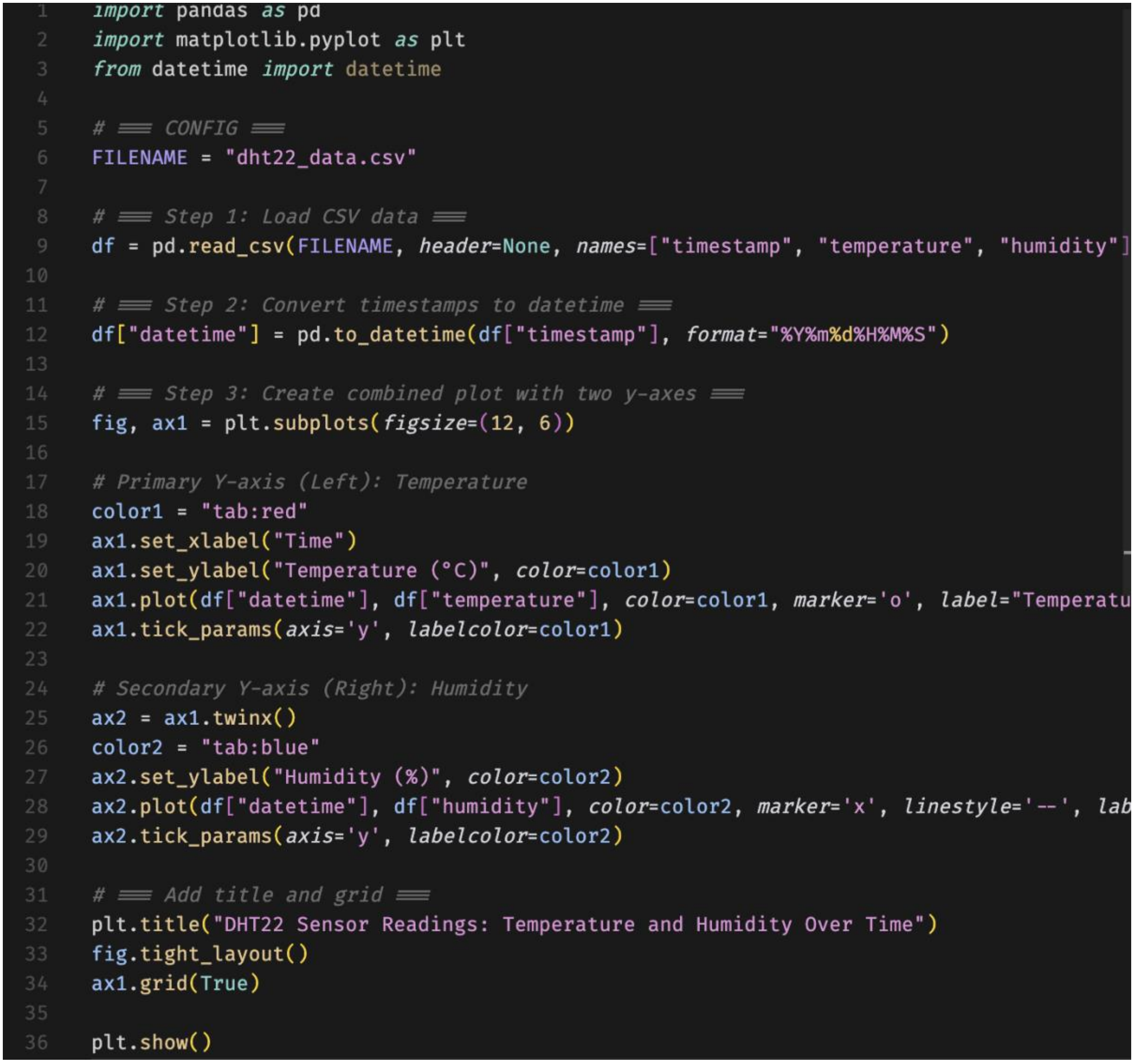


Figure 1
DHT22 Sensor Readings: Temperature and Humidity Over Time
22.2
22.0
21.8
21.6
21.4
21.2
100
95
90
85
80
75
70
65
60
Humidity (%)
07 23:45
07 23:50
07 23:55
08 00:00
08 00:05
08 00:10
08 00:15
08 00:20
Time

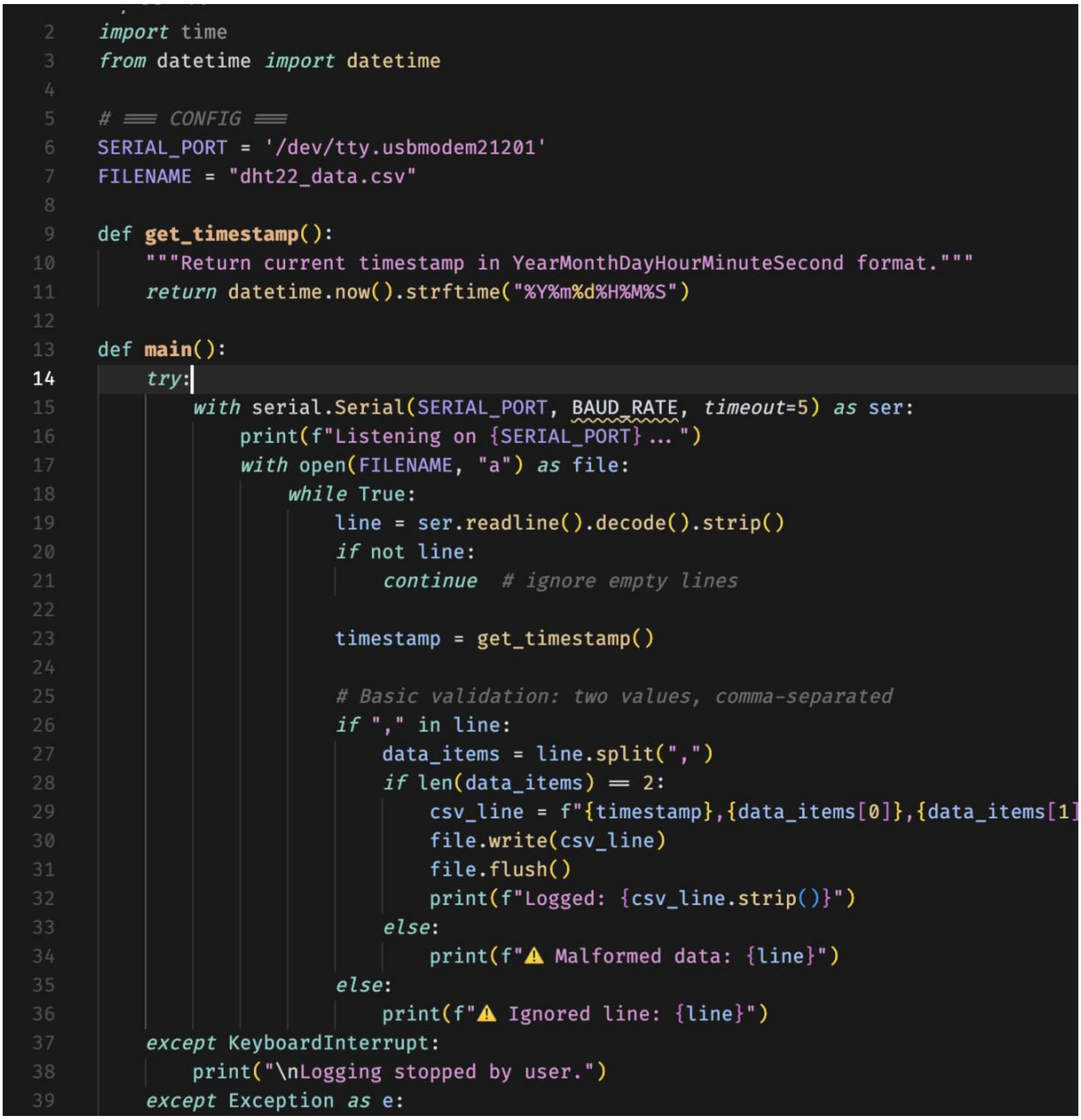
I collected temperature and humidity data using the dht22 sensor in my room, the two variables help in understanding the environmental conditions of my room. I collected new data every 10 seconds to avoid repeat data, we collected the data using the Arduino 33 nano to read the data from the dht22. The Arduino sent the temperature and humidity as comma separated values, a python script was used to receive the data and save it to a csv file.
I collected new data every 10 seconds to avoid repeat data and it is well matched with the dht22 two second update rate

a) a repeating patter I found was the temp was stable for long durations, humidity showed small fluctuations
b) Around 20250808000337 to 20250808000417, there's a rapid spike in humidity from ~ 59 to 72 to 99.8, then back down to 60.
c) when the humidity went up a slight increase in temp was noticed

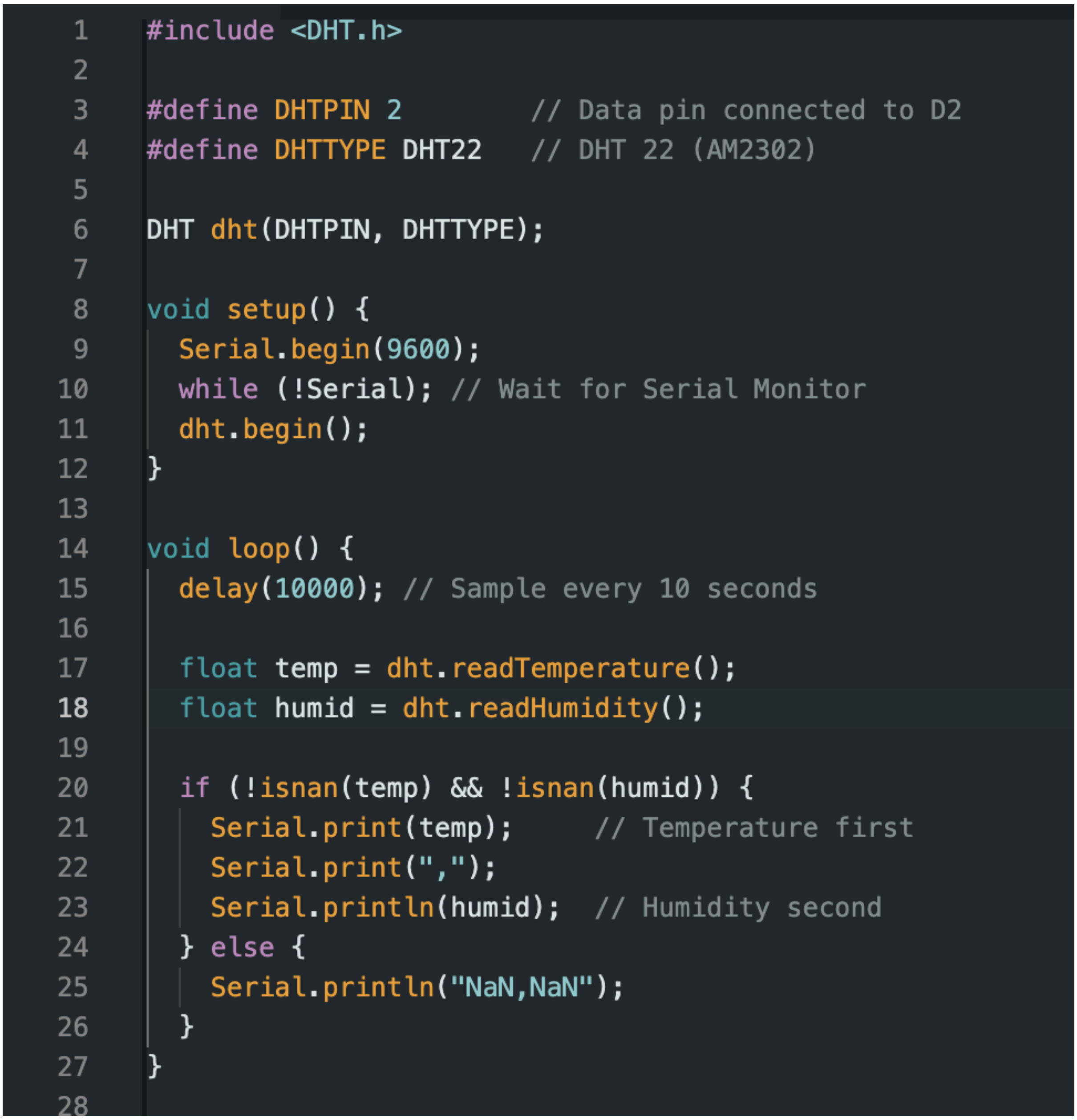
q3)



```python
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime

# === CONFIG ===
FILENAME = "dht22_data.csv"

# === Step 1: Load CSV data ===
df = pd.read_csv(FILENAME, header=None, names=["timestamp", "temperature", "humidity"]

# === Step 2: Convert timestamps to datetime ===
df["datetime"] = pd.to_datetime(df["timestamp"], format="%Y%m%d%H%M%S")

# === Step 3: Create combined plot with two y-axes ===
fig, ax1 = plt.subplots(figsize=(12, 6))

# Primary Y-axis (Left): Temperature
color1 = "tab:red"
ax1.set_xlabel("Time")
ax1.set_ylabel("Temperature (°C)", color=color1)
ax1.plot(df["datetime"], df["temperature"], color=color1, marker='o', label="Temperatu
ax1.tick_params(axis='y', labelcolor=color1)

# Secondary Y-axis (Right): Humidity
ax2 = ax1.twinx()
color2 = "tab:blue"
ax2.set_ylabel("Humidity (%)", color=color2)
ax2.plot(df["datetime"], df["humidity"], color=color2, marker='x', linestyle='--', lab
ax2.tick_params(axis='y', labelcolor=color2)

# === Add title and grid ===
plt.title("DHT22 Sensor Readings: Temperature and Humidity Over Time")
fig.tight_layout()
ax1.grid(True)

plt.show()
```



```python
import time
from datetime import datetime

# === CONFIG ===
SERIAL_PORT = '/dev/tty.usbmodem21201'
FILENAME = "dht22_data.csv"

def get_timestamp():
    """Return current timestamp in YearMonthDayHourMinuteSecond format."""
    return datetime.now().strftime("%Y%m%d%H%M%S")

def main():
    try:
        with serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=5) as ser:
            print(f"Listening on {SERIAL_PORT}...")
            with open(FILENAME, "a") as file:
                while True:
                    line = ser.readline().decode().strip()
                    if not line:
                        continue  # ignore empty lines

                    timestamp = get_timestamp()

                    # Basic validation: two values, comma-separated
                    if "," in line:
                        data_items = line.split(",")
                        if len(data_items) == 2:
                            csv_line = f"{timestamp},{data_items[0]},{data_items[1
                            file.write(csv_line)
                            file.flush()
                            print(f"Logged: {csv_line.strip()}")
                        else:
                            print(f"⚠ Malformed data: {line}")
                    else:
                        print(f"⚠ Ignored line: {line}")
    except KeyboardInterrupt:
        print("\nLogging stopped by user.")
    except Exception as e:
```



```
#include <DHT.h>

#define DHTPIN 2         // Data pin connected to D2
#define DHTTYPE DHT22    // DHT 22 (AM2302)

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  while (!Serial); // Wait for Serial Monitor
  dht.begin();
}

void loop() {
  delay(10000); // Sample every 10 seconds

  float temp = dht.readTemperature();
  float humid = dht.readHumidity();

  if (!isnan(temp) && !isnan(humid)) {
    Serial.print(temp);      // Temperature first
    Serial.print(",");
    Serial.println(humid);   // Humidity second
  } else {
    Serial.println("NaN,NaN");
  }
}
```

I implemented an Arduino sketch that reads temperature and humidity values from a DHT22 sensor every 10 seconds and sends the data as comma-separated values over the Serial port. Then, a Python script (dht22_logger.py) listens for this data, adds a timestamp using the computer's clock, and saves each reading to a CSV file in the format timestamp,temperature,humidity. A second Python script (plot_dht22_combined.py) reads this CSV file, converts the timestamps into readable datetime objects, and plots both temperature and humidity on a single graph using dual y-axes to visualize how the environment changes over time.

Q4)

https://deakin.au.panopto.com/Panopto/Pages/Sessions/List.aspx?folderID=10ec970b-fea5-4ea3-b4b7-b33200ec3313

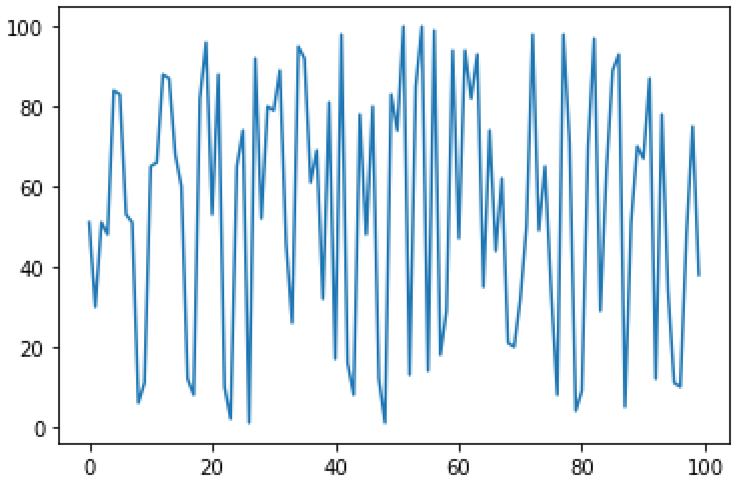
In [7]:

```
student_name = "Zakarya Guerinat"  # fill your name
student_id = "217090531"  # fill your student ID
print("Student name: " + student_name)
print("Student ID: " + student_id)
```

```
Student name: Zakarya Guerinat
Student ID: 217090531
```

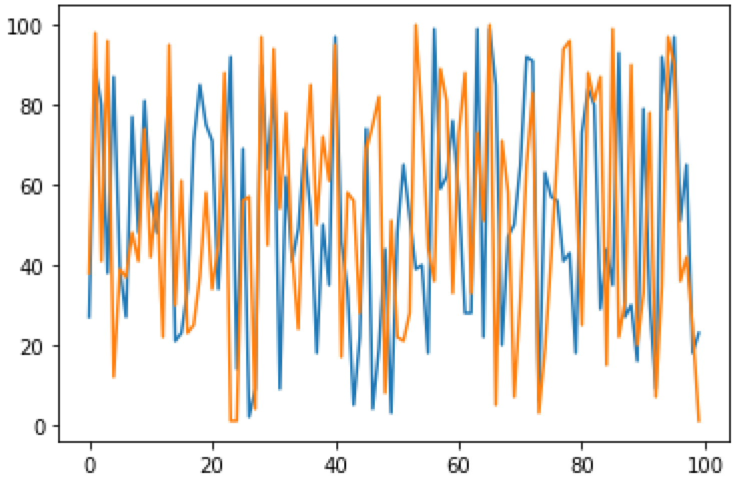
In [2]:

```
import random
import matplotlib.pyplot as plt

n_values = 100
y_values = []

# Create data (y_values) randomly between 1 and 100.
for i in range(n_values):
    y_values.append(random.randint(1, 100))

x_values = range(n_values)  # X is sequence of values 0-99
plt.plot(x_values, y_values)
plt.show()
```



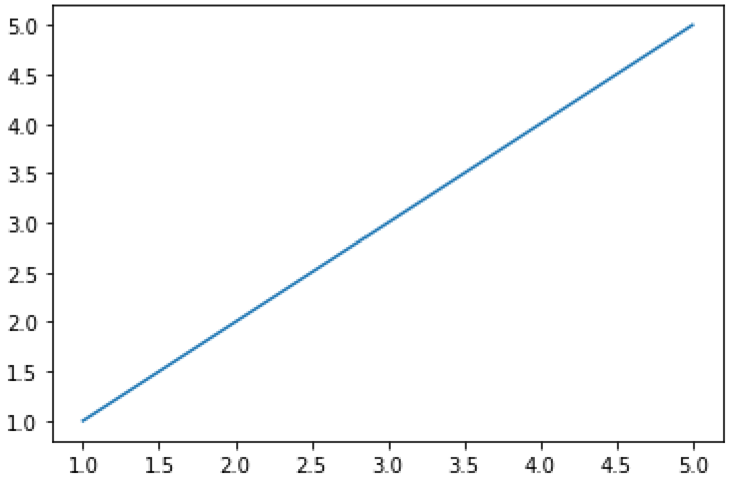
In [8]:

```
# Plot 2 variables
#

n_values = 100
y_values_1 = []
y_values_2 = []

# Create data (y_values) randomly between 1 and 100.
for i in range(n_values):
    y_values_1.append(random.randint(1, 100))
    y_values_2.append(random.randint(1, 100))

x_values = range(n_values)  # X is sequence of values 0-99
plt.plot(x_values, y_values_1)
plt.plot(x_values, y_values_2)  # call plot again draws in the same graph.
plt.show()
```



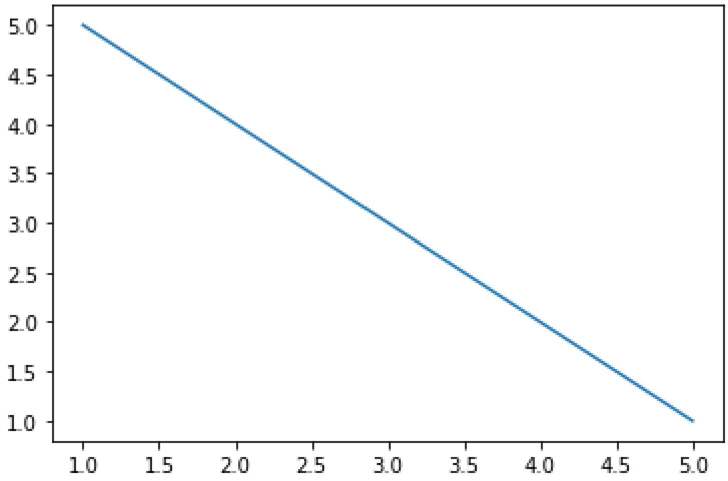
In [9]:

```
#
# Activity 1: Create data so that the plot draws an
# ascending line (y_values increase at any rate).
#

import matplotlib.pyplot as plt

x_values = [1, 2, 3, 4, 5]
y_values = [1, 2, 3, 4, 5]

plt.plot(x_values, y_values)
plt.show()
```



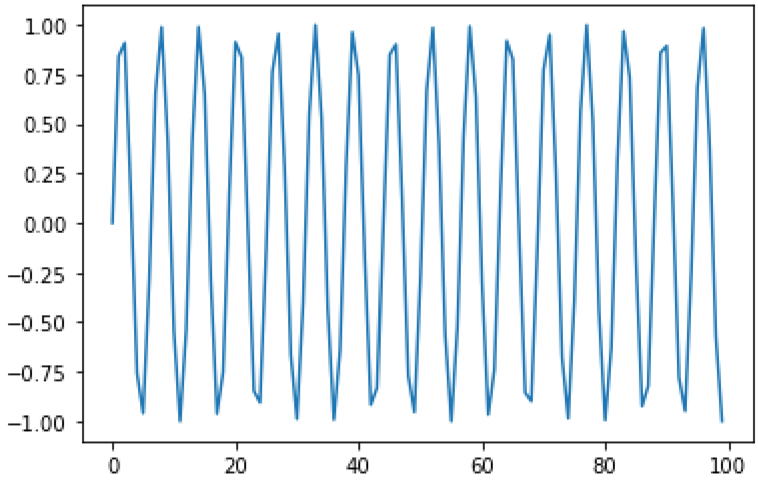
In [12]:

```
#
# Activity 2: Create data so that the plot draws a
# descending line (y_values decrease at any rate).
#

import matplotlib.pyplot as plt

x_values = [1, 2, 3, 4, 5]
y_values = [5, 4, 3, 2, 1]

plt.plot(x_values, y_values)
plt.show()
```



In [13]:

```
#
# Activity 3: Create data so that the plot draws a
# wave. You can consider using Python's math libarary, which has
# a sin function (detail https://www.w3schools.com/python/ref_math_sin.asp).
#

import matplotlib.pyplot as plt
import math

x_values = [i for i in range(100)]
y_values = [math.sin(i) for i in x_values]

plt.plot(x_values, y_values)
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
20
40
60
80
100