Task 2.1P – Working with sensors

Yasmin Pokia

S222245206

*Q1. Perform week 2 activities mentioned in the unit site and produce outputs.*

*Q2. State the hypothesis you can think out of your data. Show the graph created from the sensor data, analyse it and describe if there are any interesting patterns you can observe. Justify if your hypothesis holds, at what level; if not, then what might be the reason?*

It is hypothesized that changes in the accelerometer data will clearly identify different types of moves, for example stationary, walking, due to distinct patterns in x, y, and z axes data.

A white paper with green and orange lines

Description automatically generated

A graph showing a line of different colored squares

Description automatically generated with medium confidence

Shown above is the graph created from the LSM6dS3 module on the Arduino Nano 33 IoT accelerometer and gyroscope sensor data, over the course of approximately 35 minutes. (NOTE: these are 2 screenshots as I could not fit the whole graph at once.) As shown between minutes 0-5, 10-15, 20-25, and 30-35, the lines are flat on all three exes which indicates little to no movement. This is because I left the Arduino stationary between these times, thus the data shown is consistent. Furthermore, between minutes 5-10 there shows little patterns which indicate little movement as this is where I moved the Arduino back and forth slowly during this phase. Also, between minutes 15-20 there is more patterns which are similar to or a bit more than the first phase of movement, which indicates movement as this is where I walked up and down on the spot with the Arduino. Lastly, between the minutes 25-30, as shown there is more chaotic and less predictable patterns across all exes, indicating there is rapid movement, which is due to me randomly moving the Arduino around.

My hypothesis holds due to observing clear distinctions between different activities as expected – stationary phases showed minimal variance while movement phases showed an increase and patterned variability.

*Q3. Paste Python and Arduino sketch and explain program steps.*

**Arduino Program: (same code from GitHub from activity 2.3)**

/\*

    Ahsan Habib (habib@deakin.edu.au),

    School of Information Technology,

    Deakin University, Australia.

\*/

// 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);

}

**Arduino Sketch Explanation:**

1. Include Libraries and Declare Variables:

* #include <Arduino\_LSM6DS3.h>: contains the library needed to communicate with the accelerometer on the LSM6DS3.
* float y, z, x;: declares variables to store data from the accelerometer.

1. Setup Function:

* Serial.begin(9600);: Starts a serial connection at a rate of 9600 baud.
* while (!Serial);: Holds off on starting the serial port. This is especially helpful for boards that have USB built right in.
* The serial monitor receives a starting message from Serial.println("Started");.
* The IMU is attempted to be initialised if (!IMU.begin()). In the event that it fails, the program is essentially stopped by printing an error message and going into an infinite loop (while (1);).

1. Loop Function:

* Determines whether fresh acceleration data is available if (IMU.accelerationAvailable()).
* x, y, z; IMU.readAcceleration; records the acceleration data in x, y, and z after reading it.
* String(x) + ", " + String(y) + ", " + String(z));: Serial.println creates a string and transmits it via the serial port using the accelerometer data. This is caused by the delay(1000); and occurs every second.

**Python Program:**

import serial

import time

import csv

from datetime import datetime

import pandas as pd

import matplotlib.pyplot as plt

# Establish serial connection (adjust 'COM5' to your Arduino's port)

try:

ser = serial.Serial('COM5', 9600) # Replace 'COM5' with your actual port

print("Serial port opened successfully.")

except Exception as e:

print(f"Error opening serial port: {e}")

ser = None

if ser:

# Open CSV file to write data

with open('accelerometer\_data.csv', mode='a', newline='') as file:

writer = csv.writer(file)

# Continuously read data from Arduino

try:

print("Starting data collection...")

while True:

if ser.in\_waiting > 0:

line = ser.readline().decode('utf-8').strip()

print(f"Data received: {line}") # Debug print

timestamp = datetime.now().strftime('%Y%m%d%H%M%S')

data = line.split(', ')

if len(data) == 3: # Ensure correct data format

writer.writerow([timestamp] + data)

else:

print(f"Unexpected data format: {line}")

time.sleep(1) # Adjust sampling rate if needed

else:

print("Waiting for data...") # Debug print

time.sleep(1)

except KeyboardInterrupt:

print("Data collection stopped.")

finally:

ser.close()

print("Serial port closed.")

# Read the CSV file

data = pd.read\_csv('accelerometer\_data.csv', header=None, names=['timestamp', 'x', 'y', 'z'])

# Convert the timestamp column to a readable datetime format

data['timestamp'] = pd.to\_datetime(data['timestamp'].astype(str), format='%Y%m%d%H%M%S')

# Print the first few rows to check the conversion

print(data.head())

# Plot the data

plt.figure(figsize=(10, 6))

plt.plot(data['timestamp'], data['x'], label='X-axis')

plt.plot(data['timestamp'], data['y'], label='Y-axis')

plt.plot(data['timestamp'], data['z'], label='Z-axis')

plt.xlabel('Time')

plt.ylabel('Acceleration (g)')

plt.title('Accelerometer Data')

plt.legend()

plt.show()

**Python Script Explanation:**

1. Setup and Open Serial Connection

* Attempts to use serial to create a serial connection.Serial ( 9600, 'COM5').
* Handling of errors when opening the serial port is not possible.

1. Data Collection:

* Opens the accelerometer\_data.csv CSV file for data adding.
* Reads data from the serial buffer continuously:
* ser.readline().decode('utf-8').strip(): Removes any trailing newline or whitespace after reading one line from the serial input, decoding it from UTF-8.
* print(f)"Data received: line" ): enables debugging by printing every line of data received.
* datetime.now()   timestamp.generates a timestamp for the moment the data was received using strftime('%Y%m%d%H%M%S').
* Line is data.split(', '): Divides the line into components x, y, and z.
* Writes the data and timestamp to a CSV file.
* Pauses for a second before reading additional information (time.sleep(1)).
* Stops gathering data when a keyboard interrupt is received.

1. Read and Plot Data:

* Opens a pandas DataFrame and reads the CSV file once data collecting is complete.
* For improved graphing, converts the timestamp from a string to a datetime object.
* Plots the x, y, and z acceleration data over time using matplotlib.

*Q4. Create a video in Panopto/CloudDeakin showing your program execution, data collection and any instruction to be followed in order to run your code, share the video link here.*

<https://deakin.au.panopto.com/Panopto/Pages/Viewer.aspx?id=fbe876b4-c996-4305-9550-b1b7007dab7e>

*Q5. Create a subdirectory ‘week-2’ under directory ‘SIT225\_2024T2’, which you created for week 1 task, in your drive where you copy the Python script file, Arduino sketch file, data file and the generated graphs. Commit and push to changes to GitHub. Include the link to your repository here with a GitHub page screenshot of weekly folder content. A tutor may try to access your GitHub link, if necessary. Give access to your tutor by adding tutor’s email address as a collaborator of your private repository.*