Vibration data generation and frequency determination

Pin

- Motor
  - 5V - VCC
  - GND - GND
  - 2 - A-1B
  - 3 - A-1A
- Accelerometer
  - 3.3V - Vs
  - GND - GND
  - 5V - CS (I2S communication)
  - SDA - A4
  - SCL - A5

Arduino Code

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>

const int A1A = 3;
const int A1B = 2;
char state = 's';
int speedValue = 150;
bool keepRunning = true;

Adafruit_ADXL345_Unified accel(12345);
const unsigned long SAMPLE_INTERVAL_MS = 10;
unsigned long previousMillis = 0;

void setup() {
  Serial.begin(115200);
  if (!accel.begin()) {
    Serial.println("ADXL345 연결 실패!");
    while(1);
  }
  accel.setRange(ADXL345_RANGE_16_G);
  accel.setDataRate(ADXL345_DATARATE_100_HZ);
  Serial.println("CSV_HEADER,Timestamp(ms),X,Y,Z");

  pinMode(A1A, OUTPUT);
  pinMode(A1B, OUTPUT);
  analogWrite(A1A, 0);
  digitalWrite(A1B, LOW);
}
```

```
void loop() {
  if (!keepRunning) {
    analogWrite(A1A, 0);
    digitalWrite(A1B, LOW);
    return;
  }

  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= SAMPLE_INTERVAL_MS) {
    previousMillis = currentMillis;
    sensors_event_t event;
    accel.getEvent(&event);
    Serial.print(millis());
    Serial.print(",");
    Serial.print(event.acceleration.x, 3);
    Serial.print(",");
    Serial.print(event.acceleration.y, 3);
    Serial.print(",");
    Serial.println(event.acceleration.z, 3);
  }

  handleMotorControl();
}

void handleMotorControl() {
  if (Serial.available()) {
    String input = Serial.readStringUntil('\n');
    input.trim();

    if (input.equalsIgnoreCase("s")) {
      state = 's';
      analogWrite(A1A, 0);
      digitalWrite(A1B, LOW);
      Serial.println("Motor stopped");
    }
    else if (input.equalsIgnoreCase("c")) {
      state = 'c';
      analogWrite(A1A, speedValue);
      digitalWrite(A1B, LOW);
      Serial.println("Motor continuous mode");
    }
    else if (input.equalsIgnoreCase("q")) {
      keepRunning = false;
      analogWrite(A1A, 0);
      digitalWrite(A1B, LOW);
      Serial.println("Data collection and motor stopped by 'q'");
    }
    else {
      int val = input.toInt();
      if (val >= 0 && val <= 255) {
        speedValue = val;
        Serial.print("Speed set to: ");
        Serial.println(speedValue);
        if (state == 'c') {
          analogWrite(A1A, speedValue);
        }
      } else {
        Serial.println("Invalid speed value (0-255)");
      }
    }
  }
}
```

- Acquire accelerometer data at PWM: 100
- Motor speed can be set using a value between **0 and 255**, where a higher value indicates faster speed.
- Press **'c'** to start the motor at the specified speed.
- Press **'s'** to pause the motor.
- Press **'q'** to exit the loop (end the program).

The collected data can be saved as a **CSV file** using the Python code below.

```python
import serial
import time
from datetime import datetime

def main():
    serial_port = 'COM3'
    baud_rate = 115200

    try:
        ser = serial.Serial(serial_port, baud_rate, timeout=1)
        print(f"Connected to {serial_port} at {baud_rate} baud.")
    except serial.SerialException:
        print(f"Failed to connect to {serial_port}")
        return

    filename = f"accel_data_{datetime.now().strftime('%Y%m%d_%H%M%S')}.csv"

    with open(filename, 'w') as f:
        print(f"Logging data to {filename}")
        try:
            while True:
                line = ser.readline().decode('utf-8', errors='ignore').strip()
                if line:
                    if line.startswith("CSV_HEADER"):
                        header = line.split(",", 1)[1] if "," in line else
"Timestamp(ms),X,Y,Z"
                        f.write(header + "\n")
                        print(f"Header: {header}")
                    else:
                        f.write(line + "\n")
                        print(line)
        except KeyboardInterrupt:
            print("\nLogging stopped by user.")
        finally:
            ser.close()
            print("Serial port closed.")

if __name__ == "__main__":
    main()
```

This is the Python code for saving data to a CSV file. The file name is `save_accel_data.py`. Follow the steps below to use it:

How to Use

1. Upload the Arduino code and **close the Serial Monitor**.
2. Save the above Python code as `save_accel_data.py`.
3. In the terminal (Command Prompt), navigate to the folder where the code is located by running: `cd C:\Users\joung\Desktop\DLIP\Final Project\`
4. In the terminal, run the script using: `python save_accel_data.py`
5. Accelerometer data will be displayed in real time on the console and saved to a CSV file at the same time.
6. To stop the program, press **Ctrl + C**.

Calculate the frequency with the python code below

```python
import pandas as pd
import numpy as np
from scipy.fft import fft, fftfreq

file_path = "accel_data_20250610_220918.csv"

try:
    df = pd.read_csv(file_path, header=None)

    df.columns = ['Timestamp(ms)', 'X', 'Y', 'Z']

    # 데이터 전처리
    timestamps = df['Timestamp(ms)'].values
    x = df['X'].values.astype(float)
    y = df['Y'].values.astype(float)
    z = df['Z'].values.astype(float)

    # 가속도 벡터 크기 계산
    a_mag = np.sqrt(x ** 2 + y ** 2 + z ** 2)

    # 샘플링 주파수 계산
    time_diff = np.mean(np.diff(timestamps))  # 평균 샘플링 간격(ms)
    sample_rate = 1000 / time_diff  # Hz 단위 변환

    # FFT 분석
    N = len(a_mag)
    T = 1.0 / sample_rate
    yf = fft(a_mag - np.mean(a_mag))  # DC 성분 제거
    xf = fftfreq(N, T)[:N // 2]
    magnitude = 2.0 / N * np.abs(yf[:N // 2])

    # 최대 진동 주파수 출력
    main_freq = xf[np.argmax(magnitude)]
    print(f"▶ 주요 진동 주파수: {main_freq:.2f} Hz")

except FileNotFoundError:
    print(f"오류: {file_path} 파일을 찾을 수 없습니다.")
except Exception as e:
    print(f"오류 발생: {str(e)}")
```

Result

PWM: 80

- Frequency : 4.45 Hz
- File : accel_data_20250611_80.csv

PWM: 100

- Frequency: 6.63 Hz
- File: accel_data_20250611_100.csv

PWM: 100

- Frequency: 7.75 Hz
- File: accel_data_20250611_120.csv