Digital Signal Processing Project

-Divyansh Singhal (IMT2021522) &
Daksh Sharma (IMT2021533)

Breath rate using inertial sensors such as pressure sensor/accelerometer.

a. Choose the best sensor and its position for this activity

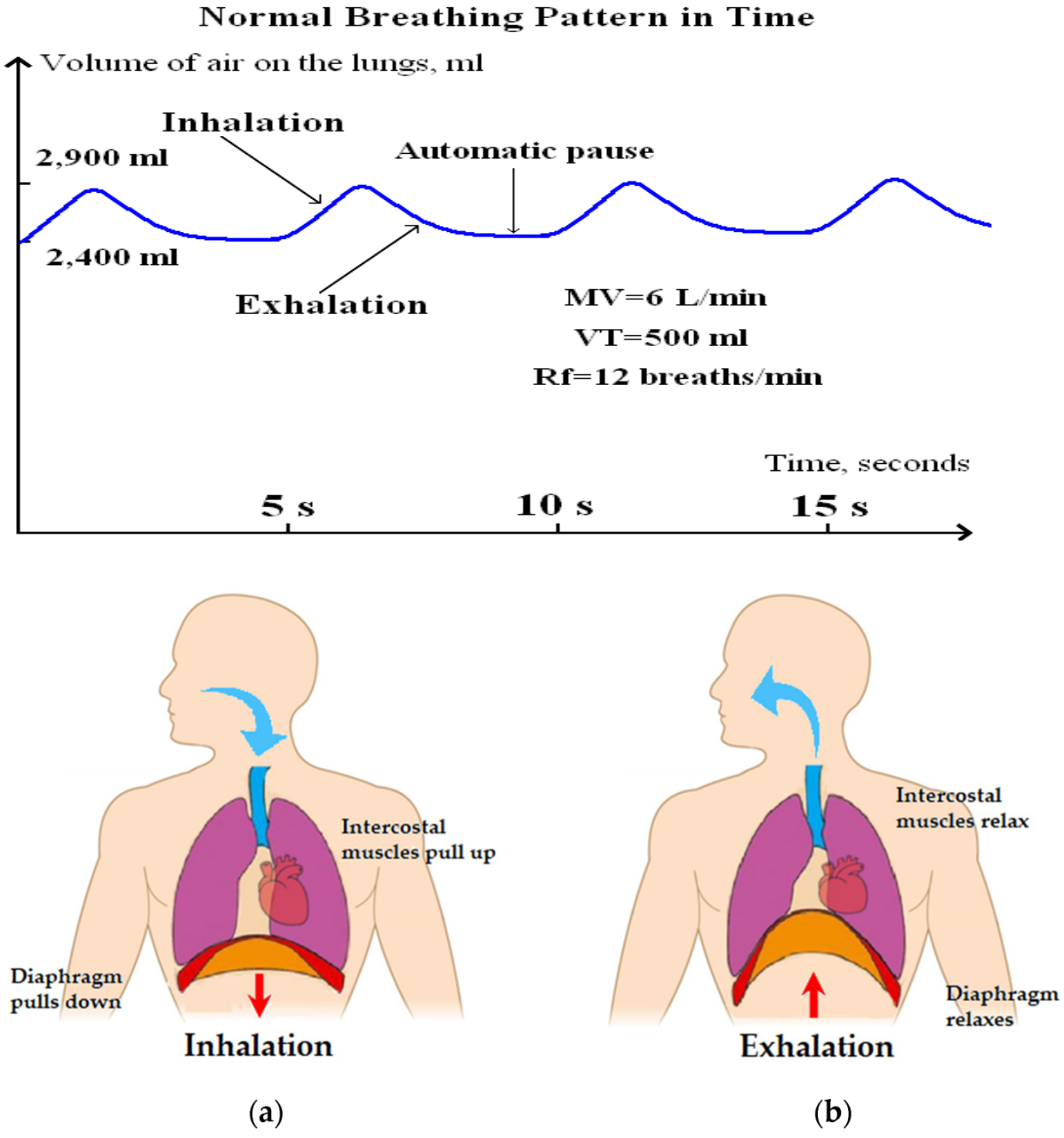
Breath rate can be measured using different types of sensors, each with its advantages and disadvantages. Two common types of sensors that can be used are **pressure sensors** and **accelerometers.**

Pressure sensors measure the changes in pressure that occur during breathing. These sensors can be placed on the chest, abdomen, or face. They are highly accurate and can provide real-time breath rate data. However, they can be uncomfortable to wear and may restrict movement.

Accelerometers measure changes in acceleration and can be used to detect changes in body movement during breathing. These sensors can be placed on the chest, abdomen, or back. They are less accurate than pressure sensors but are more comfortable to wear and can be used for continuous monitoring.

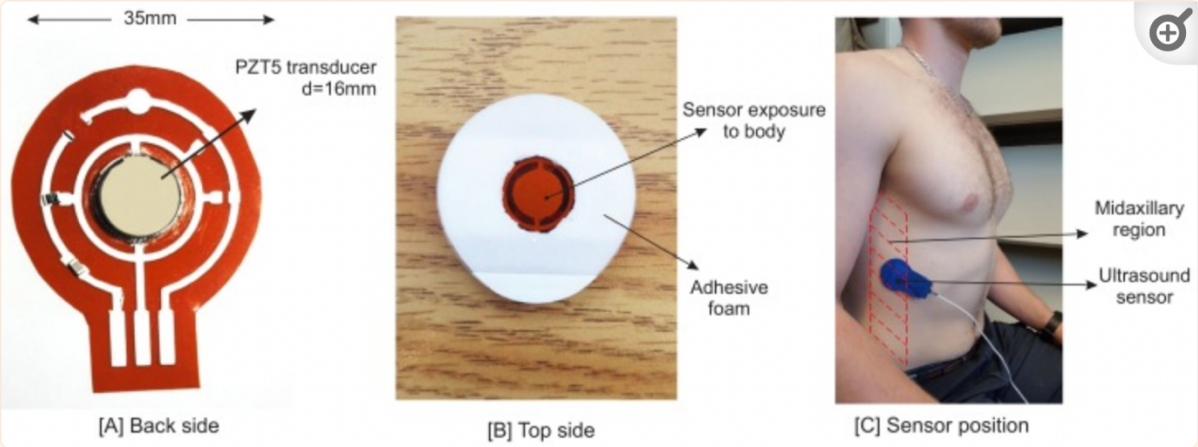
In terms of the **best sensor** and its position for measuring breath rate using inertial sensors, it depends on the specific application and user requirements. If accuracy is critical, a **pressure sensor** may be the best choice. If comfort and continuous monitoring are important, an **accelerometer** may be a better choice. Additionally, the position of the sensor will depend on the specific use case and the area of the body that best reflects changes in breathing.

Typically, **chest and abdomen** are the most common areas for sensor placement.



Normal respiration rates for an adult person at rest range from 12 to 16 breaths per minute.

Pressure Sensor



A **pressure sensor** can be used to measure breathing rate by detecting changes in pressure as a person inhales and exhales. This can be achieved by placing the sensor in close proximity to the person's nose or mouth, and measuring the changes in pressure as air flows in and out.

To interface the pressure sensor with an Arduino, you can follow the following steps:

1. Connect the pressure sensor to the Arduino as per the datasheet of the sensor. The sensor will typically have two pins for power supply, and two pins for signal output.
2. Create a new Arduino sketch and import the necessary libraries for your pressure sensor.
3. Define the pins used to connect the pressure sensor to the Arduino.
4. Set up the serial communication between the Arduino and your computer, so that the data from the pressure sensor can be transmitted to the computer.
5. In the loop() function of your Arduino sketch, read the pressure value from the sensor and send it to the serial port.
6. Use a serial monitor on your computer to view the pressure readings.

Graphs:

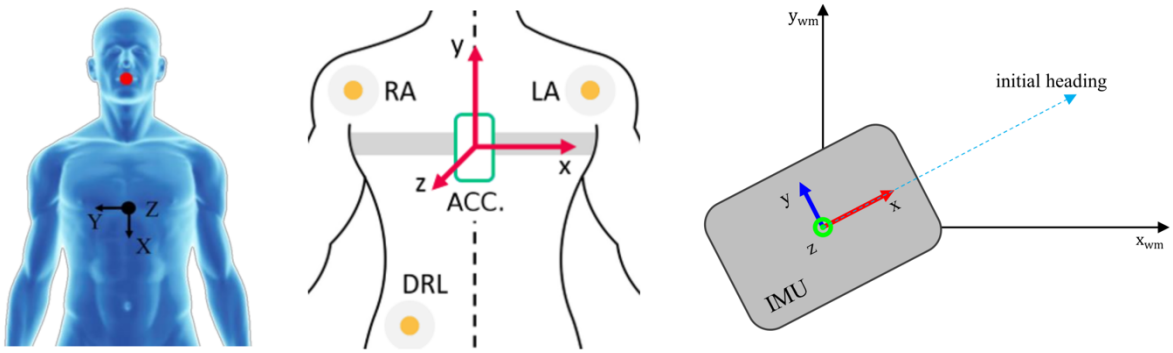

value 1
Interpolate
STOP
Type Message
SEND
New Line
Search


Interpolate
STOP
6173
6185
6197
SEND
New Line
Search

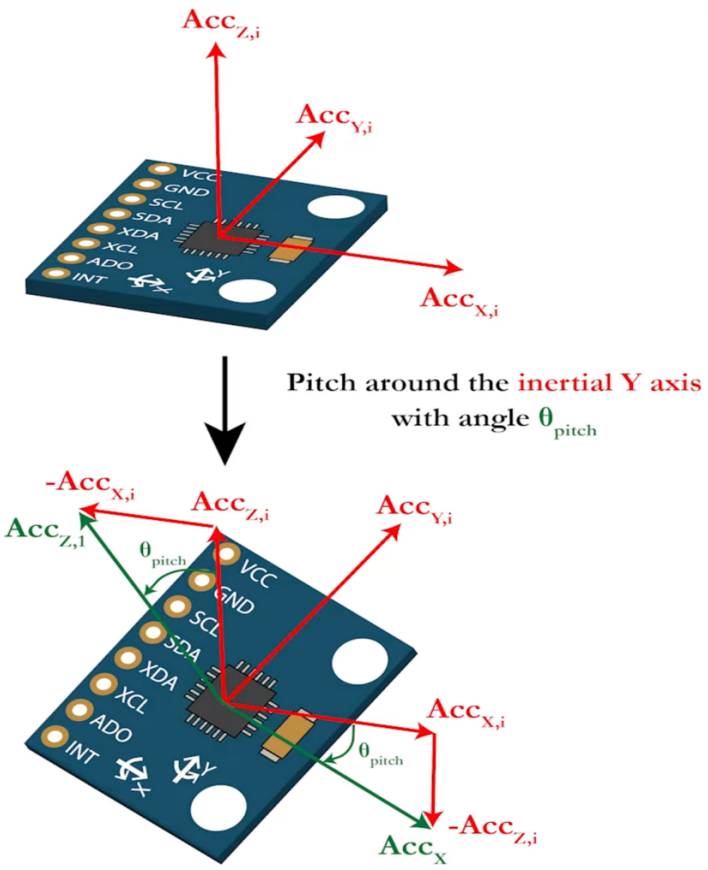
Code:

```
#include <Wire.h>
#include <Adafruit_MPL115A2.h>

#define pressurePin A0

void setup() {
  Serial.begin(9600);
}

void loop() {
  int pressureValue = analogRead(pressurePin);
  Serial.println(pressureValue);
  delay(1000);
}
```

Accelerometer



(a) (b) (c)

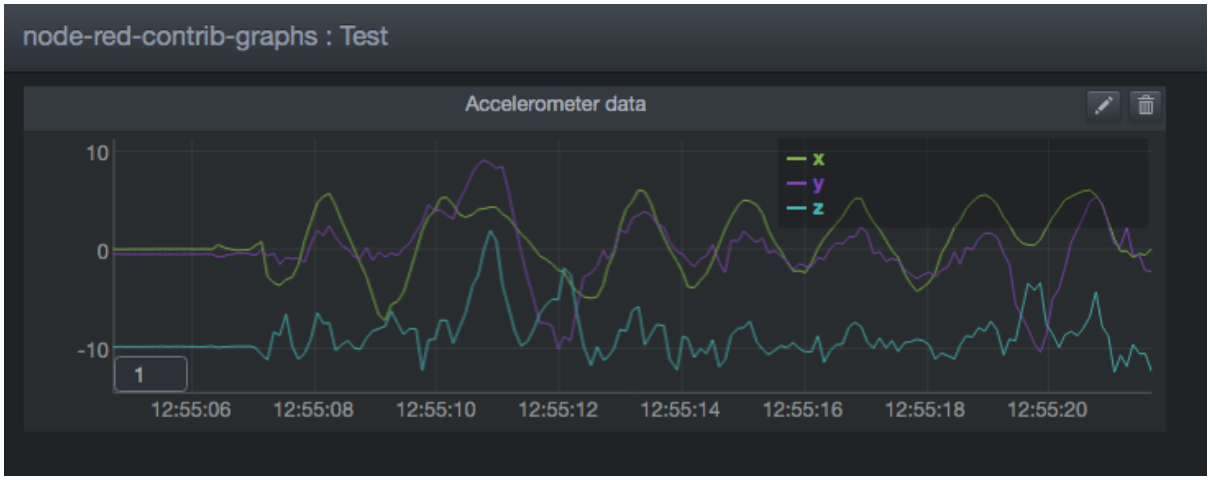


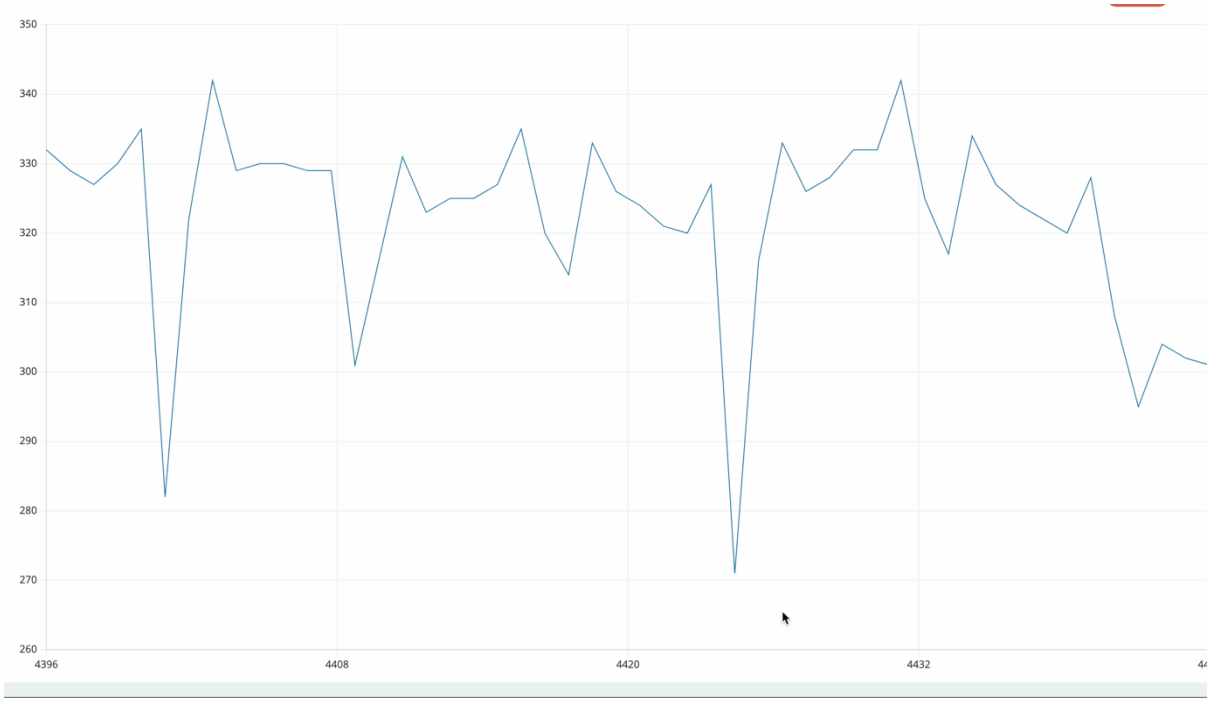
The **MPU6050 is a 6-axis gyroscope and accelerometer sensor** that can be used to measure breathing rate indirectly by detecting the movement of the chest during inhalation and exhalation. This can be achieved by placing the sensor on the chest or abdomen, and measuring the acceleration and angular velocity as the person breathes.

To interface the MPU6050 sensor with an Arduino, you can follow the following steps:

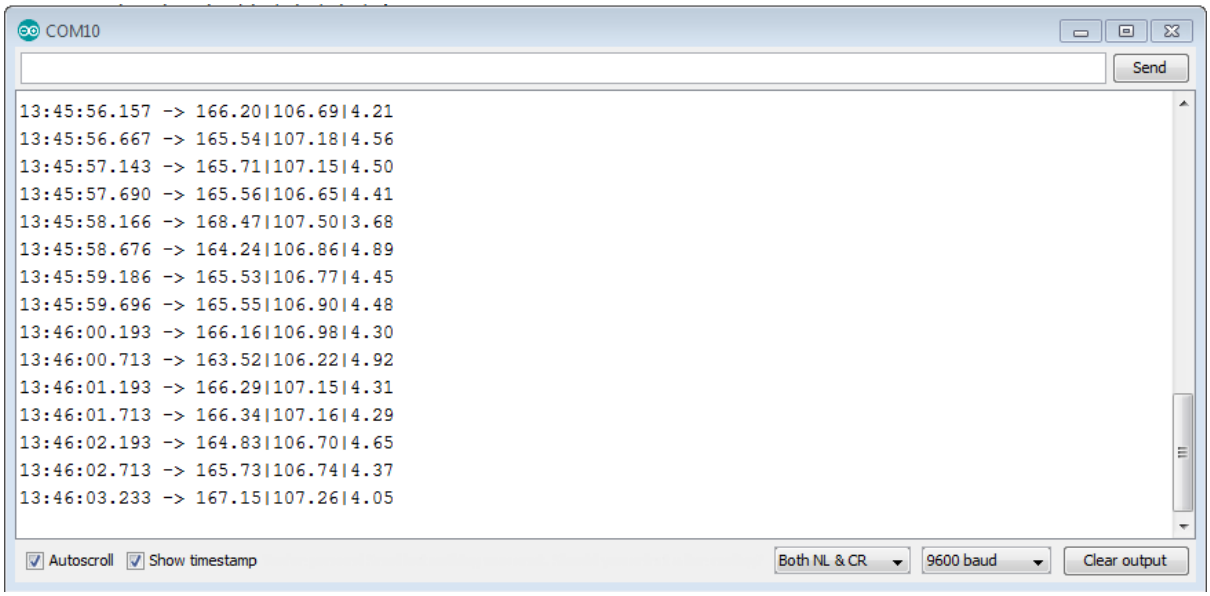
1. Connect the MPU6050 sensor to the Arduino as per the datasheet of the sensor. The sensor will typically have two pins for power supply, two pins for I2C communication, and several pins for signal output.
2. Create a new Arduino sketch and import the necessary libraries for your MPU6050 sensor.
3. Define the pins used to connect the MPU6050 sensor to the Arduino.
4. Set up the I2C communication between the Arduino and your MPU6050 sensor, so that the data from the sensor can be transmitted to the Arduino.
5. In the loop() function of your Arduino sketch, read the acceleration and angular velocity values from the MPU6050 sensor and use them to calculate the breathing rate.
6. Use a serial monitor on your computer to view the breathing rate readings.

Graph:





Output:



Code:

```arduino
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;
void setup() {
  Wire.begin();
  Serial.begin(9600);
  mpu.initialize();
}

void loop() {
  Vector3f accel = mpu.readAccel();
  Vector3f gyro = mpu.readGyro();

  float xAccel = accel.x;
  float yAccel = accel.y;
  float zAccel = accel.z;
  float xGyro = gyro.x;
  float yGyro = gyro.y;
  float zGyro = gyro.z;

  // Calculate breathing rate using acceleration and angular velocity values
  // ...

  Serial.print("Breathing rate: ");
  Serial.print(breathingRate);
  Serial.println(" breaths per minute");

  delay(1000);
}
```

b. Analyse the signal

Once the signal is acquired, it can be analyzed in various ways. Some of the common methods for analyzing a breathing rate signal include:

1. **Frequency domain analysis:** This involves converting the time-domain signal into the frequency domain using Fourier transform and analyzing the frequency components of the signal. The breathing rate can be estimated by identifying the dominant frequency component in the signal.

```
#include <FFT.h>
#define SAMPLES 1024
#define SAMPLING_FREQ 100
#define BREATHING_RATE_MIN 0.05
#define BREATHING_RATE_MAX 2
int data[SAMPLES];
double spectrum[SAMPLES];
double breathingRate;
void setup() {
  Serial.begin(9600);
}
void loop() {
  // acquire data from sensor and store it in data array
  for (int i = 0; i < SAMPLES; i++) {
    data[i] = readSensor();
    delay(10);
  }
  // compute spectrum using FFT
  FFT.Windowing(data, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
  FFT.Compute(data, spectrum, SAMPLES, FFT_FORWARD);
  FFT.ComplexToMagnitude(spectrum, SAMPLES);
  // find dominant frequency component in spectrum
  int maxIndex = 0;
  double maxValue = 0;
  for (int i = 1; i < SAMPLES/2; i++) {
    if (spectrum[i] > maxValue) {
      maxValue = spectrum[i];
      maxIndex = i;
    }
  }
  // calculate breathing rate from dominant frequency component
  if (maxIndex > 0 && maxIndex < SAMPLES/2) {
    breathingRate = (double) maxIndex * SAMPLING_FREQ / SAMPLES;
    if (breathingRate > BREATHING_RATE_MIN && breathingRate < BREATHING_RATE_MAX) {
      Serial.print("Breathing rate: ");
      Serial.print(breathingRate);
      Serial.println(" breaths per minute");
    }
  }
  delay(1000);
}
int readSensor() {
  // read data from sensor and return value
}
```

2. **Peak detection:** This involves detecting the peaks in the signal and calculating the time interval between them. The breathing rate can be estimated by dividing the number of peaks by the duration of the signal.

```
#define THRESHOLD 500
#define MIN_PEAK_DISTANCE 200
#define BREATHING_RATE_MIN 0.05
#define BREATHING_RATE_MAX 2

unsigned long lastPeakTime = 0;
unsigned int peakCount = 0;
double breathingRate;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = readSensor();

  // detect peak in signal
  if (sensorValue > THRESHOLD && (millis() - lastPeakTime) > MIN_PEAK_DISTANCE) {
    lastPeakTime = millis();
    peakCount++;
  }

  // calculate breathing rate from peak count
  breathingRate = (double) peakCount * 60 / (millis() / 1000.0);
  if (breathingRate > BREATHING_RATE_MIN && breathingRate < BREATHING_RATE_MAX) {
    Serial.print("Breathing rate: ");
    Serial.print(breathingRate);
    Serial.println(" breaths per minute");
  }

  delay(10);
}

int readSensor() {
  // read data from sensor and return value
}
```

3. **Statistical analysis:** This involves calculating statistical measures such as mean, variance, and standard deviation of the signal, and using them to estimate the breathing rate.

```
#define SAMPLES 100
#define BREATHING_RATE_MIN 0.05
#define BREATHING_RATE_MAX 2

int data[SAMPLES];
double breathingRate;

void setup() {
  Serial.begin(9600);
}

void loop() {
  // acquire data from sensor and store it in data array
  for (int i = 0; i < SAMPLES; i++) {
    data[i] = readSensor();
    delay(10);
  }

  // calculate mean and standard deviation of data
  double mean = 0;
  for (int i = 0; i < SAMPLES; i++) {
```

4. **Machine learning:** This involves training a machine learning model on a dataset of breathing rate signals and using it to predict the breathing rate from new signals.

The choice of analysis method depends on the nature of the signal and the requirements of the application. For example, if the signal is noisy and has multiple frequency components, frequency domain analysis may be more appropriate. On the other hand, if

the signal is relatively clean and has distinct peaks, peak detection may be a simpler and more accurate method.

c. Extract the breath rate from the signal

To extract the breath rate from the signal, you would need to identify the peaks and troughs in the waveform corresponding to each breath. One way to do this is to use a peak detection algorithm to locate the highest points in the waveform, which represent the peak of inhalation.
Then, you can use the time between each peak as the period of each breath, and calculate the breath rate as the inverse of the period.
Alternatively, you could use a Fourier transform to analyze the frequency content of the signal and identify the dominant frequency corresponding to the breath rate.

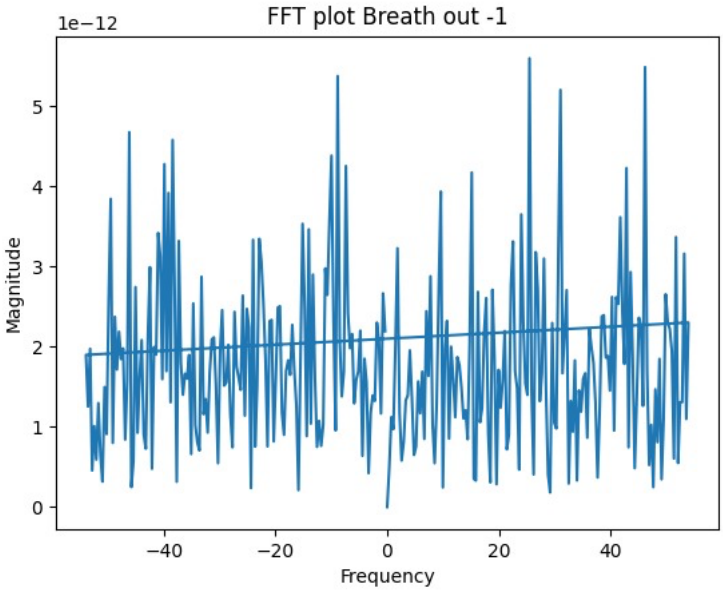
The method for extracting the breathing rate from a signal depends on the type of signal being used.

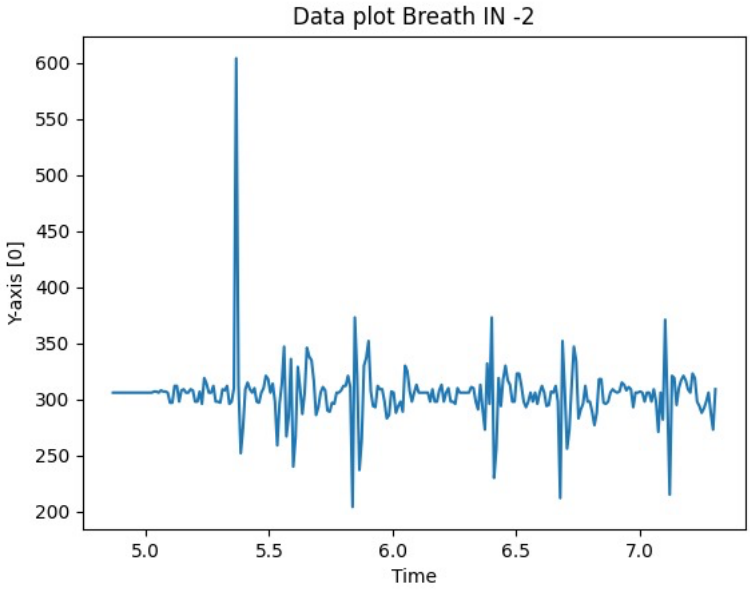
1. **Pressure signal:**

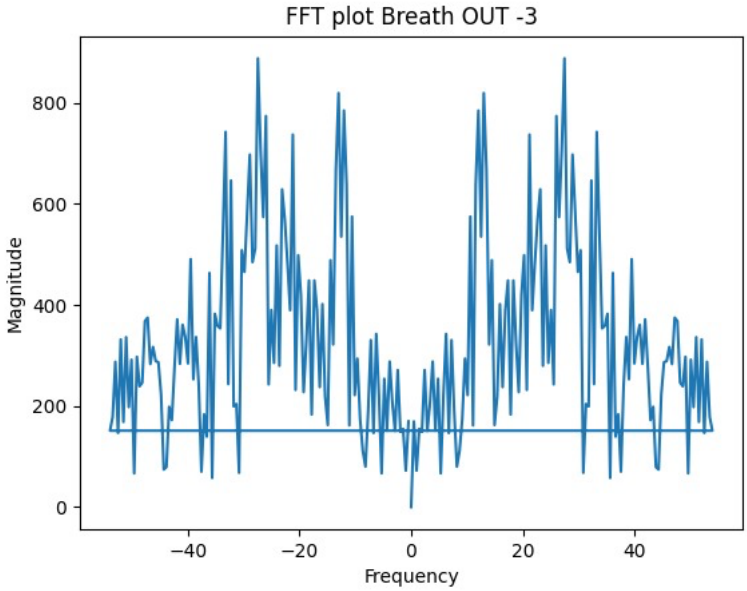
A pressure signal can be obtained from a pressure sensor placed on the chest or abdomen. To extract the breathing rate from the pressure signal, you can use peak detection. The peaks in the signal correspond to the inhalation and exhalation phases of breathing, and the time interval between them can be used to calculate the breathing rate.

2. **Flow signal:**

A flow signal can be obtained from a flow sensor placed in the breathing circuit. To extract the breathing rate from the flow signal, you can use peak detection or frequency domain analysis. The peaks in the signal correspond to the inhalation and exhalation phases of breathing, and the dominant frequency component in the spectrum can be used to estimate the breathing rate.


FFT plot Breath out -1
1e−12
Magnitude
Frequency
0
1
2
3
4
5
−40
−20
0
20
40


Data plot Breath IN -2
Y-axis [0]
Time
200
250
300
350
400
450
500
550
600
5.0
5.5
6.0
6.5
7.0


FFT plot Breath OUT -3
800
600
400
200
0
Magnitude
−40
−20
0
20
40
Frequency