Title:

Design and Analysis of a Flame Detection System Based on Signal Behavior Analysis

Prepared by:

Eng. Hussain Hafedh Almasabi

Date:

January 2025

Table of Contents

Contents

1. Introduction

Fire detection is an important safety application used to reduce damage and protect lives.
Traditional flame sensors can detect fire, but they may produce false alarms in some situations.
In this project, a flame detection system based on signal analysis and machine learning is presented.
The flame sensor signal is collected and processed to extract simple features such as mean, standard deviation, RMS, and ratio.
These features are then used to train a Random Forest classifier to distinguish between different flame conditions, including normal state, transient flame, and real fire.
The main goal of this project is to study the behavior of the flame sensor signal and evaluate how machine learning can improve fire detection accuracy.
The system is implemented and tested using offline data analysis, with future improvements proposed for real-time embedded implementation.

2. Sensor and Signal Model

➢ 2.1 Flame Sensor Overview

The flame sensor is a commonly used real-world sensor for detecting the presence of fire.
It is capable of providing both an analog output and a digital output indicating low or high detection levels.
The analog output represents a continuous voltage signal
that varies with flame intensity.
In this project, the focus is placed on the analog signal rather than the digital output this choice allows the signal behavior to be observed and analyzed instead of relying on simple binary detection. See figure 1.

.

Figure 1: Flame sensor module used in the experimental setup

Esp32 Role in the system

The Esp32 was used as the main microcontroller in this project.

It was responsible for reading the analog output from the flame sensor and collecting the row data in real time.
The collected sensor values were then transmitted to the computer, were signal processing, feature extraction, and machine learning classification were performed using Python. See figure 2.1.1

Figure 2.1.1 shows the Esp32 microcontroller

➢ 2.2 physical principle of operation

Fire emits infrared (IR) radiation as a result of the combustion process.
The flame sensor detects this infrared radiation and converts it into an electrical voltage signal.
The magnitude of the output voltage increases as the intensity of the detected infrared radiation increases.
The electrical voltage is an analog signal, which represents the continuous behavior of the flame is the primary signal of interest in this work.

➢ 2.3 output signal characteristics

The flame sensor data were collected under three different operating conditions: baseline (no fire), transient flame (small fire), and sustained fire (real fire). For each condition, the sensor output was recorded as a function of time.

The recorded data were then processed and visualized using MATLAB.
A separate time-domain figure was generated for each operating condition to illustrate the temporal behavior of the flame sensor signal.

The resulting figures provide a clear comparison between the three cases and highlight the differences in signal characteristics associated with each flame condition.

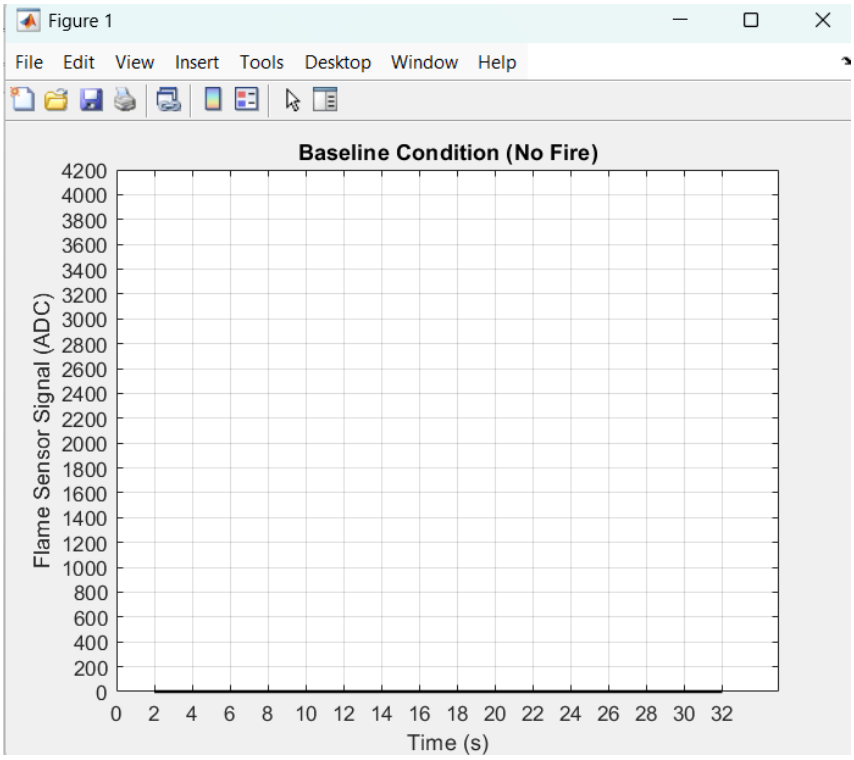


Figure 1: baseline flame sensor recorder in the absence of the fire.

Figure 1 shows the flame sensor output under baseline conditions with no fire present. The signal remains close to zero throughout the observation period, indicating stable ambient conditions and low infrared radiation.

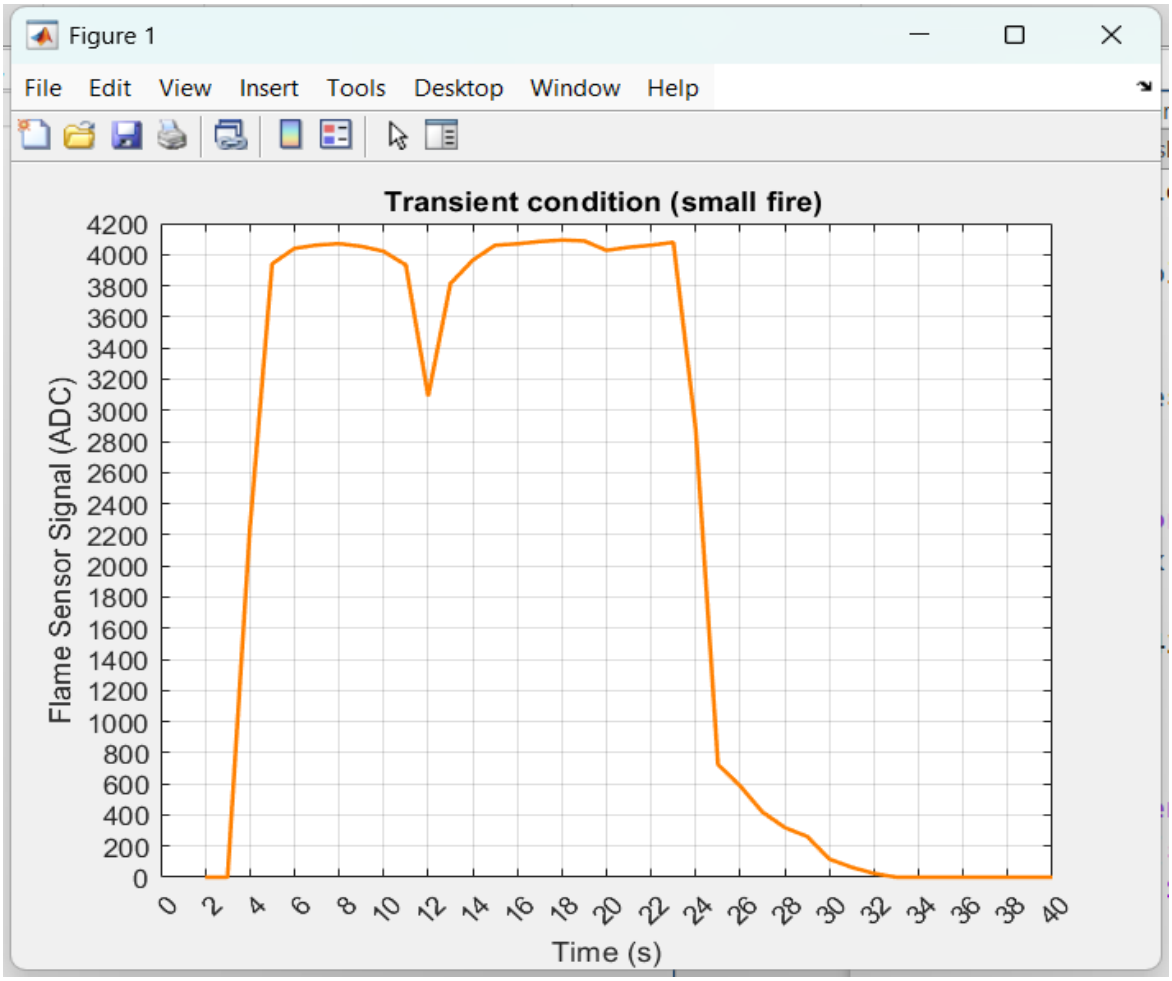


Figure 2: Transient flame (small fire) sensor signal.

Figure 2 presents the sensor response to a transient flame source. A rapid increase in the signal is observed during flame exposure, followed by a decay phase once the flame is removed. This behavior reflects the short-lived nature of small fire events, However we will see in figure 3 the real fire case which the flame sensor signal stays high over time with some value changes which is exactly the behavior of the real fire.

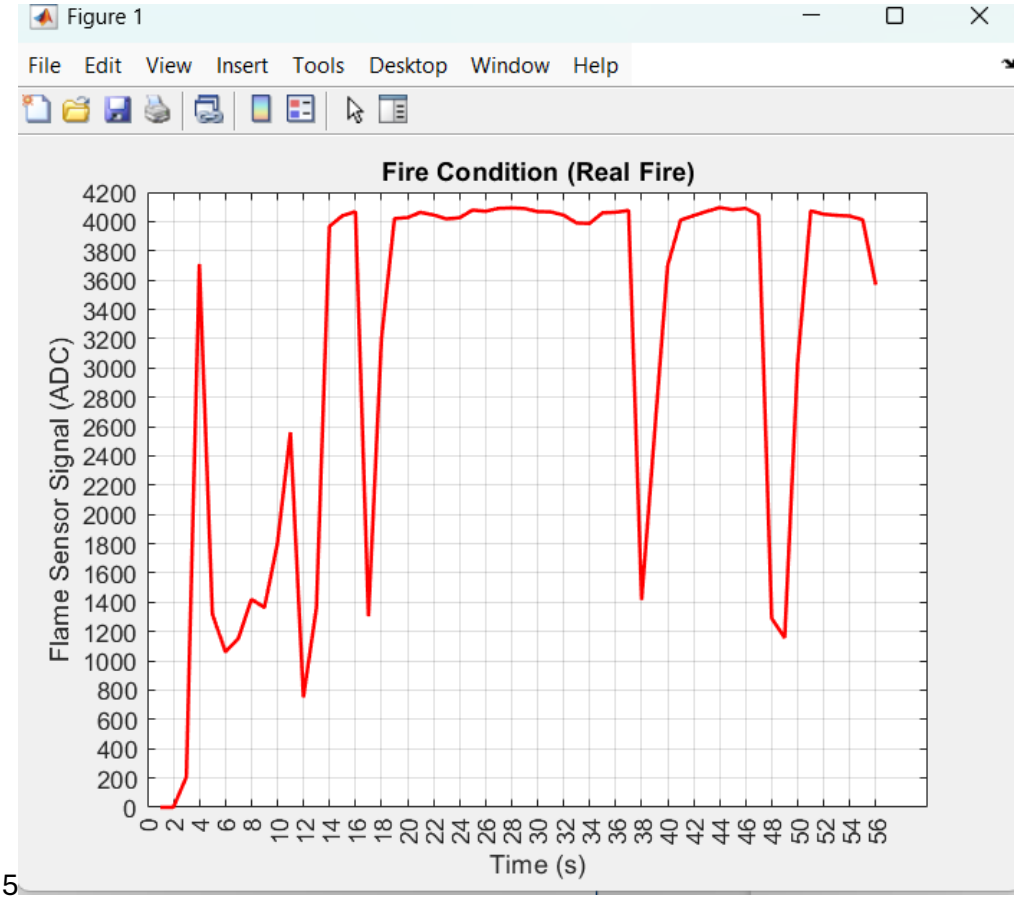


Figure 3: fire flame (Real Fire) sensor signal.

Figure 3 shows the signal starting with a high rise, followed by some fluctuations in the signal value, which indicates the presence of a real fire. In addition, the signal remains continuous and maintains high values over time.

➢ 2.4 Sampling frequency

Fs = 1/Ts is called sampling frequency (cycles per second or Hz) or sampling rate (samples per second).
In this project we used Matlab to compute Fs as shown in figure 2.4.1 and 2.4.2

```
1   baseline = readtable("baseline.csv"); % read all values from file
2   t_baseline = baseline.t_s; % make a vector with all values of t_s
3   fs_b = 1/mean(diff(t_baseline)); % fs = 1/Ts diff: computes the difference betw
4
5   fire = readtable("fire.csv");  % read all values from file
6   t_fire = fire.t_s; % make a vector with values of t_s
7   fs_f = 1/mean(diff(t_fire)); % fs = 1/Ts diff: computes the difference between
8
9
10  transient = readtable("transient.csv");  % read all values from file
11  t_transient = transient.t_s; % make a vector with values of t_s
12  fs_t = 1/mean(diff(t_transient)); % fs = 1/Ts diff: computes the difference bet
```

Figure 2.4.1: Matlab code calculates sampling frequency

```
>> untitled2
>> fs_b

fs_b =

     1

>> fs_f

fs_f =

    0.9818

>> fs_t

fs_t =

     1
```

Figure 2.4.2 :
sampling frequency values

small variation in the estimated sampling frequency in figure 2.4.2 are due to serial communication and processing delay, while the overall sampling rate remains approximately constant.

The estimated sampling frequency will be used in frequency domain analysis to determine the frequency resolution and interpret the FFT results.

3. frequency domain analysis

There are many benefits to make the signal in frequency domain like..

- Understand the frequencies inside the signal
- Filtering
- Feature extraction
- System analysis
- Detection of the vibration

➢ 3.1 key parameters

N : number of samples

Δf : frequency resolution

- for baseline condition :

N = 30 fs = 1

Δf = fs/N = 1/30 = 0.03 Hz

- For transient condition :

N = 38 fs = 1

Δf = fs/N = 1/38 = 0.0263 Hz

- For fire condition:

N = 54 fs = 1

Δf = fs/N = 1/54 = 0.0185 Hz

The reason of the difference between the values of Δf is the the different recording duration I used for each condition.

➢ 3.2 Discrete Fourier Transform

This is the way to convert the signal from time domain to frequency domain

we can do that mathematically but now we are not focusing on theoretical calculations so we will depend on FFT which is the fastest way and the most useful way in our computers.

➢ 3.3 Fast Fourier Transform (FFT) implementation

Why we are focusing on FFT?

1- FFT is an efficient algorithm to compute DFT,
2- Reduces computational complexity
    - From N^2 to N/2 log(N)
3- This makes FFT practical for real signals and large data sets

➢ 3.4 Zero Padding

1- FFT works most efficiently when number of samples N = 2^m

2- in this project, recorded signals do not always satisfy this condition

3- therefore, zero padding was applied to extend the signal length to the nearest power of two

4- zero padding:

- does not change the original signal
- does not add new information
- improves frequency resolution and spectrum visualization

➢ 3.5 FFT Analysis of flame signal

1- Apply FFT to:

- Baseline signal:

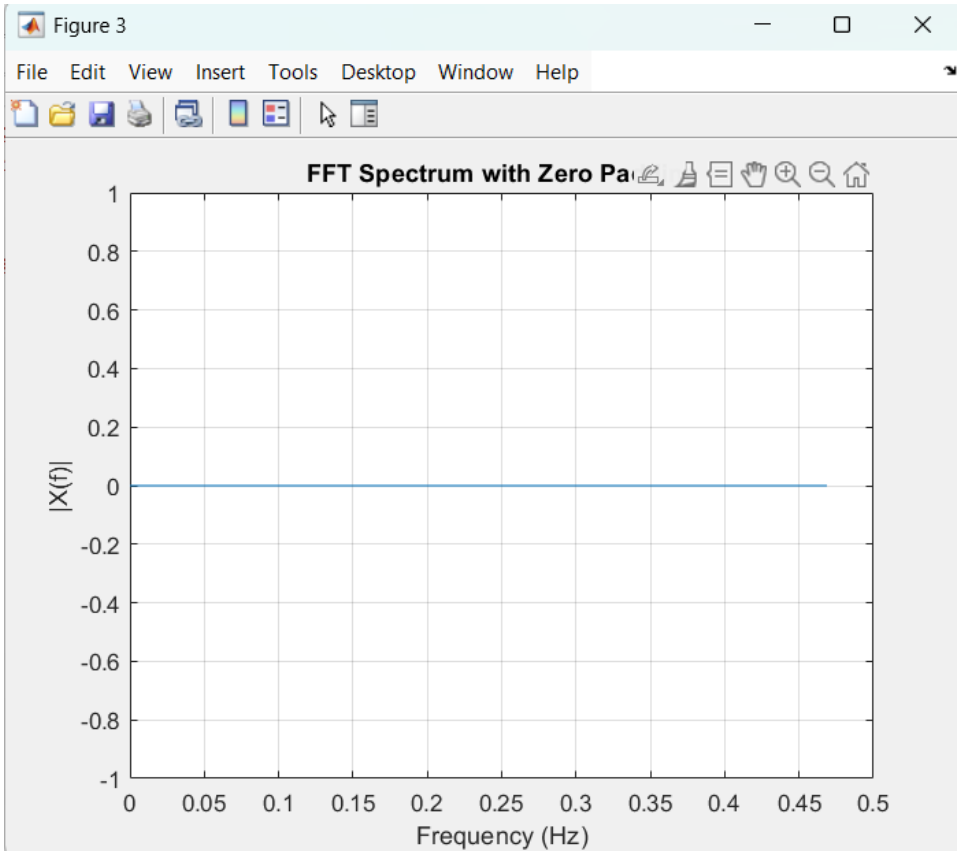


Figure 3.5.1 Frequency domain (baseline condition)

The baseline FFT shows no dominant frequency components (the frequency at the highest |X(t)|), which is excepted in the absence of fire.

- Transient signal:

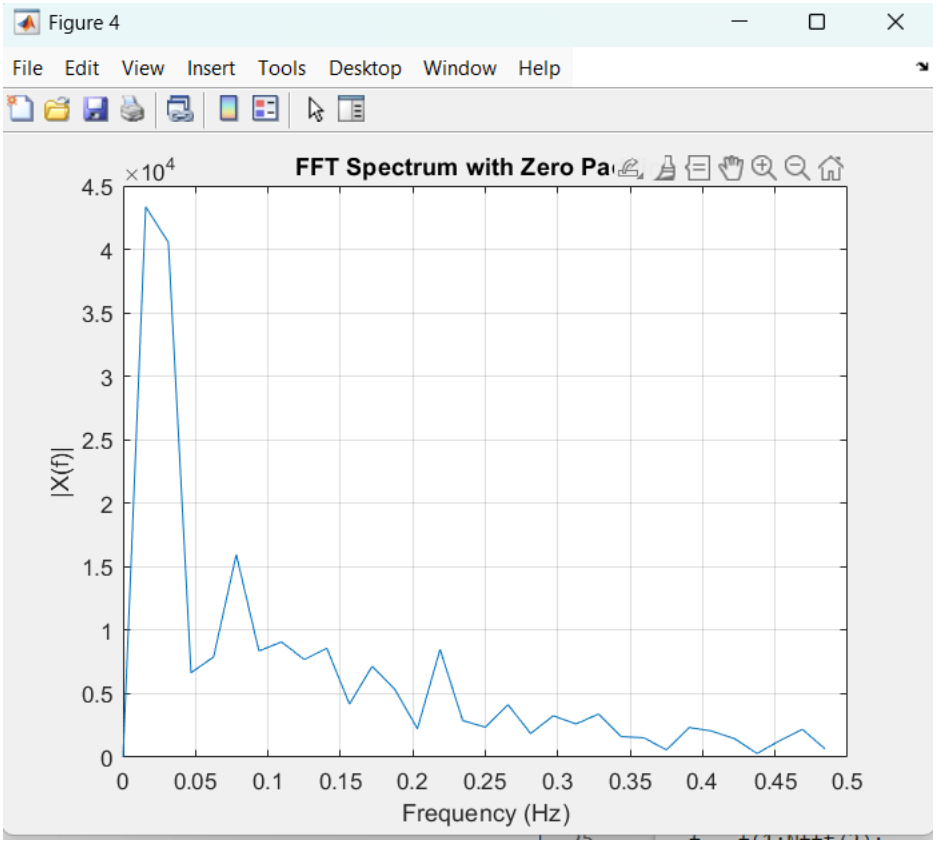


Figure 3.5.2 frequency domain (transient condition)

The transient FFT shows high energy concentrated at low frequencies, followed by a gradual decrease as frequency increases. This behavior reflects the short-duration and non-periodic nature of transient fire events, such as a match flame.

- Fire signal:

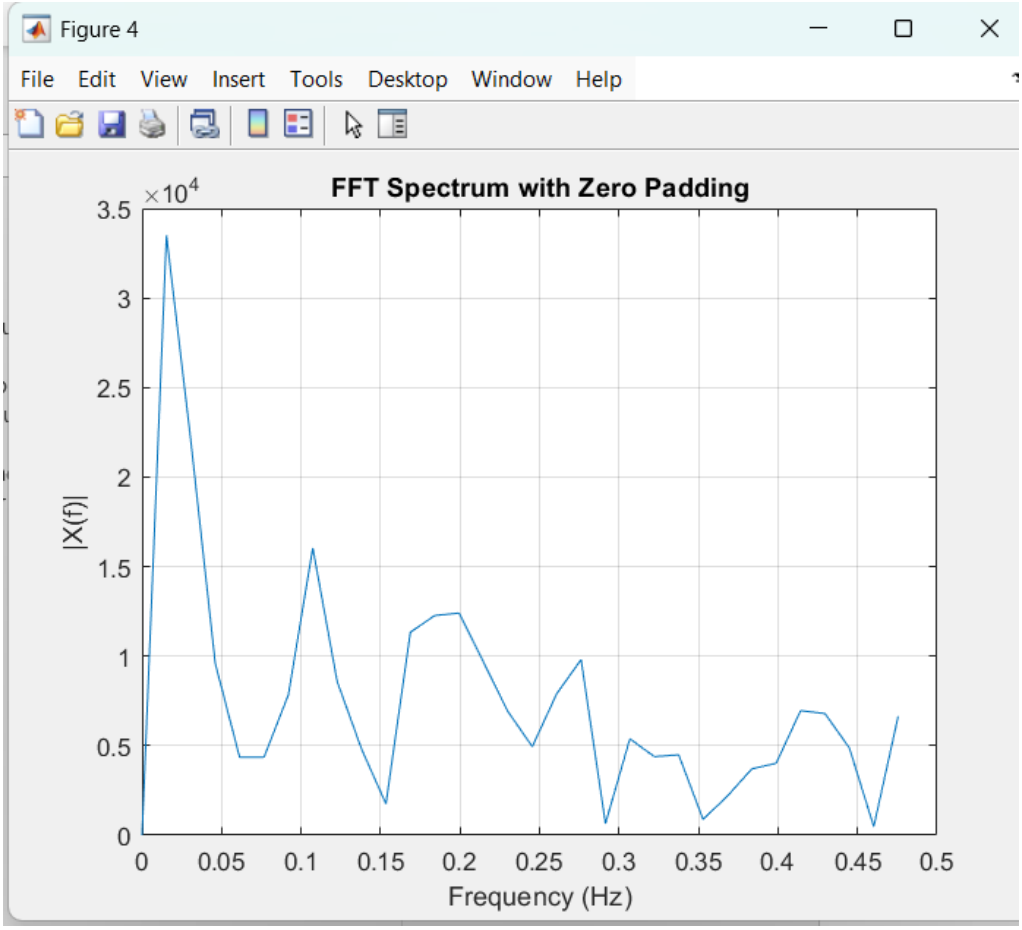


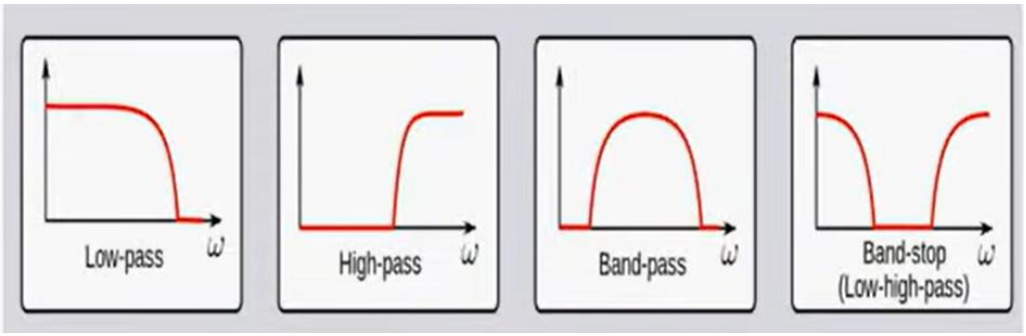
Figure 3.5.3 frequency domain (Fire condition)

The fire FFT shows strong low-frequency components with energy spread over time.

4. Filtering

- **Definition**

  Filtering in DSP a filter is a device or process some unwanted components or features on a signal by partially or completely suppressing the frequency band, let us take some examples of filters..



Low pass filter allows the low frequencies to pass through and thereby filtering out the high frequencies.

High pass filter allows the high frequencies to pass through and thereby filtering out the low frequencies.

Band [ass filter allows a certain band to pass through and thereby filtering the rest of the frequencies.

Band stop filter filters out a certain band off frequencies and thereby allowing the rest of the frequencies pass through.

4.1 FIR Low-Pass Filter Desing

as shown in section **3.5** the real fire signal has dominant energy at low frequencies, based on this observation, **a low-pass filter** is the most suitable choice for this condition.

4.2 FIR Filter Implementation

- **Baseline condition:**

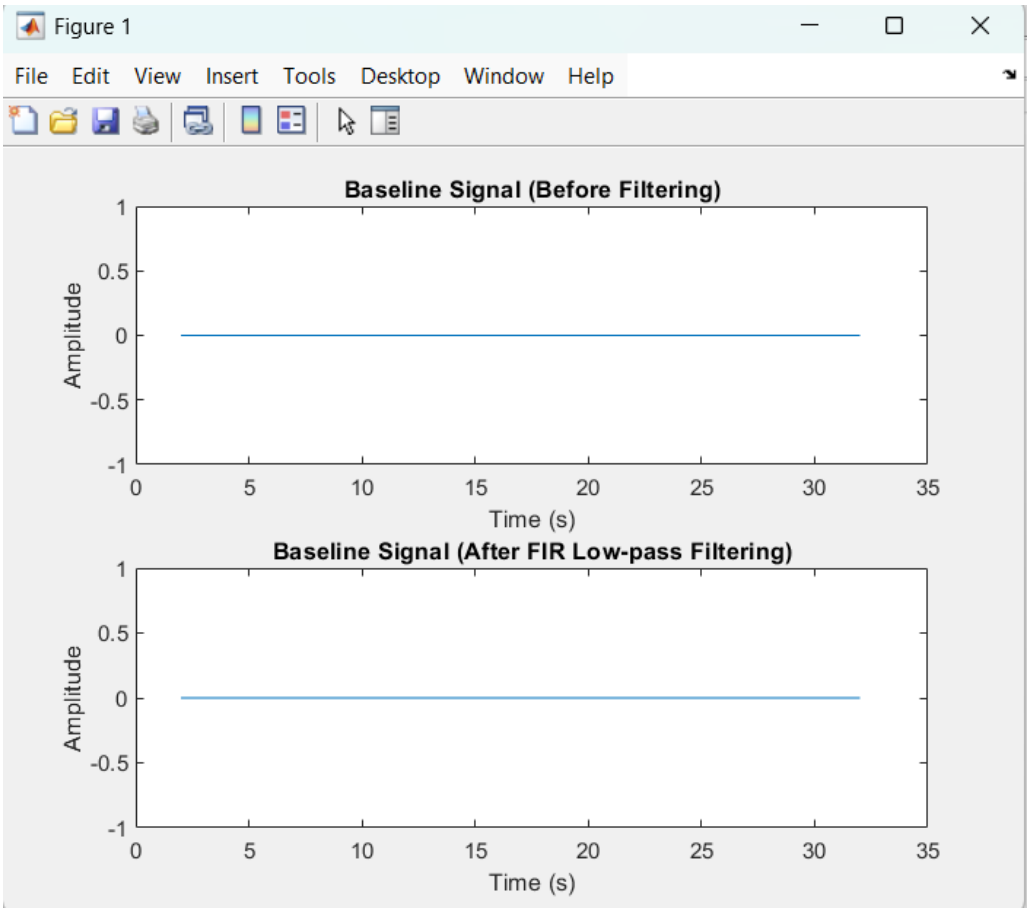


Figure 4.2.1 shows the baseline signal before and after filtering

- **Transient condition:**

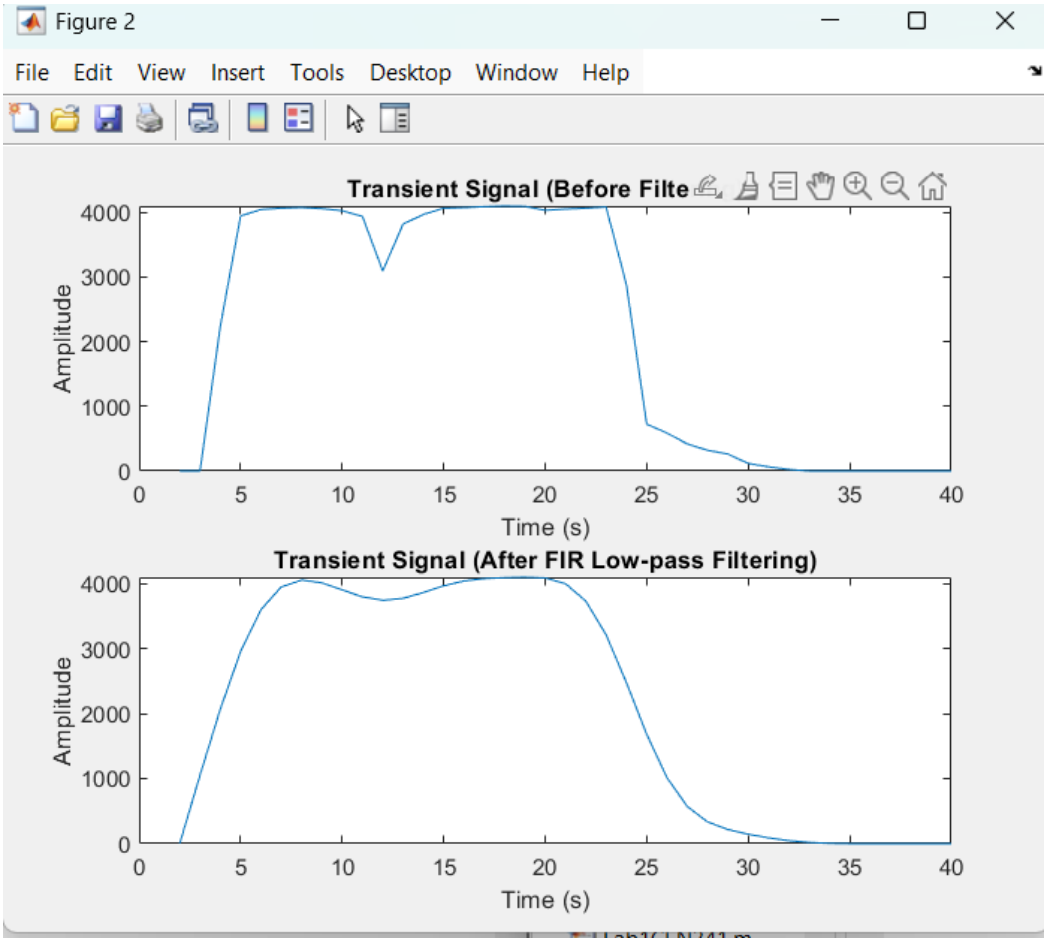


Figure 4.2.2 shows the transient signal before and after filtering

As shown in Figure 4.2.2 FIR filters introduce a linear phase delay, the signal becomes smoother and high-frequency noise is removed.

- **Fire condition:**

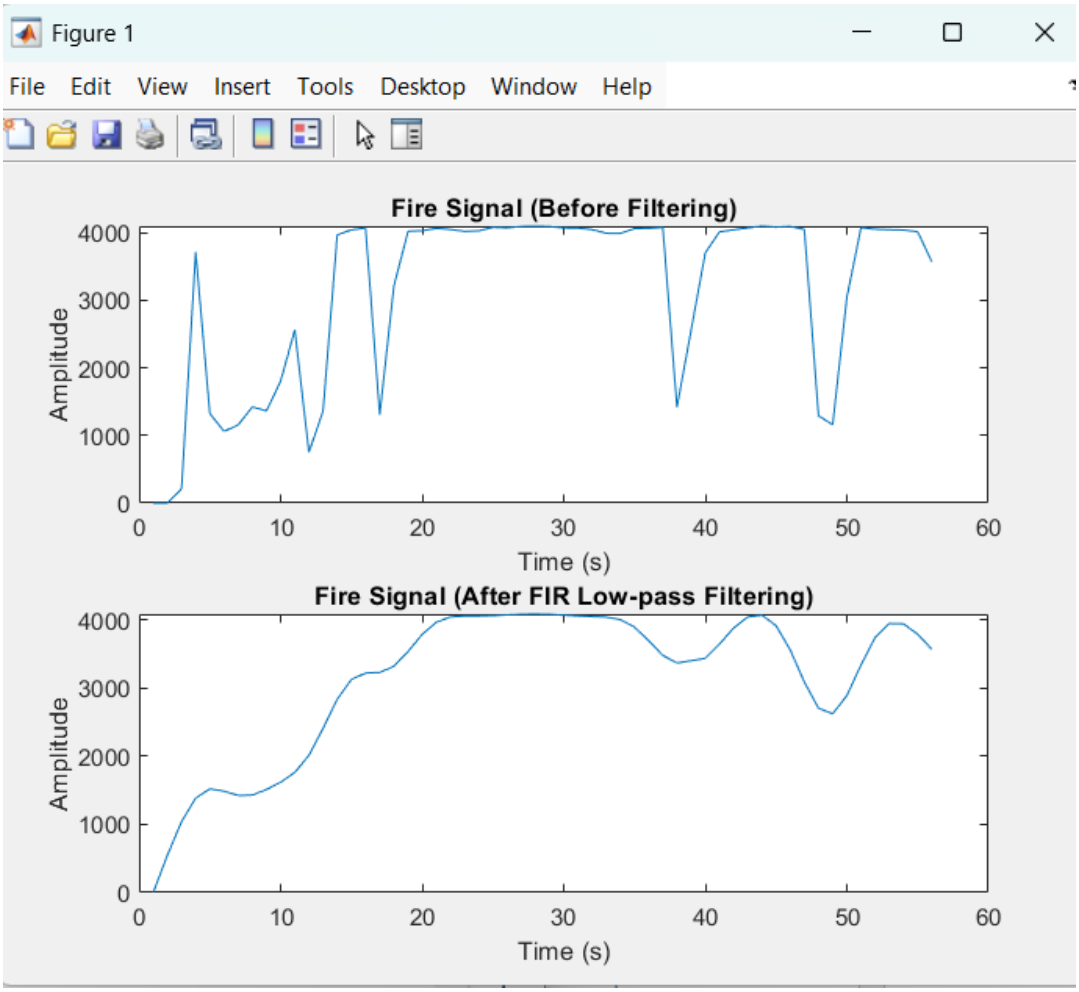


Figure 4.2.3 shows the Fire signal before and after filtering

As shown in Figure 4.2.2 FIR filters introduce a linear phase delay, the signal becomes smoother and high-frequency noise is removed.

5. Feature Extraction

➢ 5.1 Motivation

After filtering the signal, feature extraction is applied to convert time-domain signals into meaningful numerical descriptors that can be used for classification and decision making.

➢ 5.2 Extracted Features

- **Mean value:**

The mean value represents the average amplitude of the signal over time.It provides a simple indication of the overall signal level under each condition. We use the Matlab function mean() to calculate the average value.

Baseline(No Fire): the signal stays close to zero, so the mean value is approximately zero.

Mean = 0

Transient: the mean value is lower than the Fire condition because the signal gradually decreases toward zero over time.

Mean = 2129.8

Fire condition: the mean value is the highest among all conditions, because the signal maintains a high amplitude over time.

Mean = 2539.8

- **Standard Deviation (STD)**

The standard deviation measures how much the signal fluctuates around its mean value. It provides an indication of the signal stability over time. We use the Matlab function std() to calculate the standard deviation.

STD close to the mean ⟹ the signal has noticeable variation

STD much lower than the mean ⟹ the signal is more stable

Baseline(No Fire): the signal remains close to zero with minimal fluctuations, resulting in standard deviation close to zero

Transient: the standard deviation is relatively high compared to the mean value, indicating noticeable variations as the signal gradually decreases over time. Std = 1872.5

Ratio = std/mean = 1872.5/2129.8 = 0.879

Fire condition: although the mean value is the highest, the standard deviation is relatively lower than the transient condition, which indicates a more stable high-amplitude signal. Std = 1627.4

Ratio = std/mean = 1627.4/2539.8 = 0.641

- RMS (Root Mean Square)

RMS represents the effective energy of the signal over time.
We use the Matlab function rms () to calculate the Root Mean Square.

Higher RMS value ⟹ stronger and more persistent signal activity.

Baseline(No Fire): the signal remains close to zero with minimal energy, resulting in Root Mean Square close to zero.

Transient: the transient condition exhibits noticeable energy over a limited time duration. Therefore, the RMS value is higher than the baseline condition but lower than the fire condition due to the temporary nature of the signal, RMS = 2820

Fire condition: the fire condition exhibits noticeable energy over a long duration. Resulting in the highest RMS value among all conditions.
This indicates strong and persistent signal activity. RMS = 3008.5

Thus, RMS is reliable feature for distinguishing between transient and fire conditions in the classification stage.

➢ **Feature vector using window = 5**

In the section we want to increase the samples to get all the possible states and that's helps us for the classification model

Feat = [mean std rms ratio]

- Baseline

```
0   0   0   0
0   0   0   0
0   0   0   0
0   0   0   0
0   0   0   0
0   0   0   0
```

- Transient

```
1933.5   1444.0   2325.2    1.3
3939.8    100.6   3940.8   39.2
3874.2    125.7   3875.8   30.8
4066.8     40.9   4067.0   99.5
2421.4   1102.8   2614.6    2.2
 271.0    190.9    320.3    1.4
  13.7     19.6     22.2    0.7
```

- Fire

```
898.4   627.4   1059.2     1.4
1493.9   78.8   1495.6    19.0
2428.4  563.5   2480.2     4.3
3417.1  242.3   3424.0    14.1
4035.6   40.5   4035.8    99.6
4076.5    6.3   4076.5   644.7
4077.0   68.0   4080.2    58.9
3523.2  139.6   3525.4    25.2
3892.0  203.4   3896.2    19.1
2927.4  291.2   2939.0    10.1
3797.0  156.5   3799.6    24.3
```

➢ 5.3 Preparation for the classification

In this section, six data samples were collected from the sensor for each condition to build a reliable data set for fire classification using a **Random Forest Model.**

To ensure robustness against ambient lighting conditions, three samples were recorded with the room lights turned off, and three samples were recorded with the room lights turned on.

Only the baseline condition is illustrate, where the phone flashlight intentionally used to simulate ambient lighting noise.

- baseline condition with flashlight

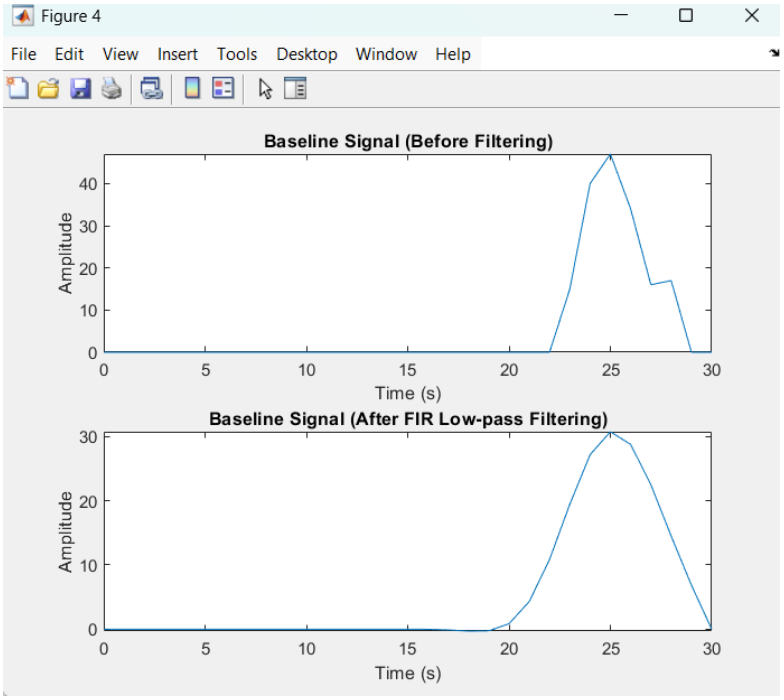


Figure 5.2.1 shows the baseline signal with slight noise caused by ambient lighting simulated using the phone flashlight.

➢ 5.4 Feature vector for each sample

To increase the number of training samples and improve the performance of the classifier, a windowing was applied to the filtered signal.
Each signal was divided into fixed-length windows of 5 seconds.
For each window, four features were extracted:

- Mean value
- Standard deviation (STD)
- Root mean square (RMS)
- Mean-to-standard-deviation ratio

The extracted feature vectors were organized into a structured data set and saved in CSV format.
Each row represents one windowed sample, while each column represent a specific feature.
The data set is included with the project files.

6. Random Forest Classification

In this section, a machine learning model was used to classify flame sensor signals into different fire conditions.

The goal of this part is to check if the extracted can help distinguish between:

- No fire (Baseline)
- Transient fire
- Real fire

A **Random Forest** classifier was chosen because It is simple, reliable, and works well with small datasets.

6.1 Dataset preparation

The dataset was created using the features extracted in the previous section.

Feature_vector = [mean STD RMS ratio] // ratio between STD and Mean

6.2 Model setup

The **Random Forest Model** was implemented using **Python** and the **scikit-learn** library.

The main parameters used in the model are:

- Number of trees: 500
- Class weight: balanced
- Minimum samples per leaf: 2
- Random state: 42

Class balancing was applied to reduce the effect of unequal data distribution between classes.

6.3 Training and Results

The dataset was split into training and testing sets.

After training the model, the classification accuracy reached approximately 73%.

This result shows that the extracted features are useful for identifying flame behavior, even with a limited dataset.

6.3.1 screenshot of the accuracy and confusion matrix

```
Accuracy:
73.53%

Confusion Matrix:
[[ 8  0  0]
 [ 2  6  2]
 [ 0  5 11]]
```

6.4 Discussion

The accuracy is affected by several factors:

- Small dataset size
- Similar signal behavior between Transient and Real fire.

7.Limitations

This project was implemented within limited time and resources.

The dataset size is relatively small and was collected under controlled conditions, which may affect the overall accuracy of the system.

In addition, the current system depends mainly on flame sensor readings only.

8.Proposed Improvements

Several improvements can be applied to enhance the accuracy and reliability of the system.

8.1 Increasing window size

In the current implementation, a small time window was used for feature extraction.

One possible improvement is to increase the window size from a small value (e.g., 5 sec) to a larger window (e.g., 15 sec).

A larger window can:

- Provide more stable feature values
- Improve classification accuracy

8.2 Increasing sensor readings

Another improvement is to increase the number of readings collected from the flame sensor.

8.3 Adding Temperature Sensor (DS18B20) as an Additional feature

The system can be improved by integrating a **Temperature sensor**.

Temperature readings can be used as additional features alongside flame sensor features.

This sensor fusion approach can:

- Improve fire detection reliability
- Reduce false positives
- Help distinguish between real fire and false flame-like signals

8.4 Real-Time Embedded Implementation

Future work may include implementing the trained model on embedded hardware such as ESP32

This allows real-time fire detection and makes the system suitable for practical applications.